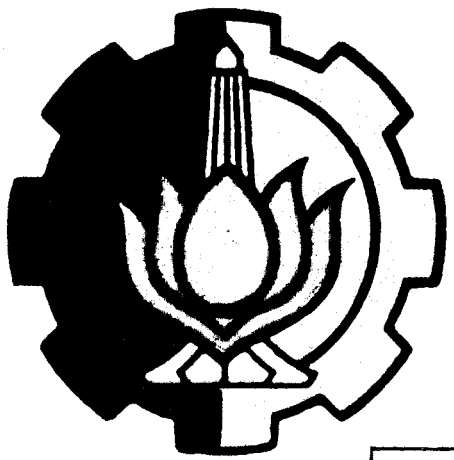
PEMBACA ANGKA TULISAN TANGAN DENGAN ALGORITMA JARINGAN SARAF TIRUAN MENGGUNAKAN TRANSPUTER T805



OLEH:

ARIS HANDOKO

NRP : 288 220 0944

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1994

PEMBACA ANGKA TULISAN TANGAN DENGAN ALGORITMA JARINGAN SARAF TIRUAN MENGGUNAKAN TRANSPUTER T805

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar
Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. HARMANI SUHARDJO)	(Ir. HENDRA KUSUMA)

SURABAYA
AGUSTUS, 1994

ABSTRAK

Dalam tugas akhir ini dipelajari dan diaplikasikan Jaringan Saraf Tiruan dalam bentuk alat untuk membaca karakter angka yang ditulis dengan tangan. Seperti sifat manusia, Jaringan Saraf Tiruan dapat belajar dari pengalaman. Jaringan Saraf Tiruan dapat mencari sendiri fungsi transfer dengan jalan mempelajari berbagai contoh yang diberikan.

Implementasi Jaringan Saraf Tiruan untuk pengenalan pola karakter angka ini dilakukan menggunakan Transputer T805. Transputer T805 merupakan suatu komputer dalam satu chip dimana di dalamnya sudah terdapat prosesor, unit floating-point, link komunikasi, memori internal dan interface memori eksternal. Pelaksanaan proses belajar dan pemakaian Jaringan Saraf Tiruan maupun interface dengan scanner dilakukan oleh transputer. Sementara PC bertugas melayani kebutuhan transputer baik input dari keyboard, akses ke disket maupun tampilan ke layar monitor.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, yang hanya karena rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul :

PEMBACA ANGKA TULISAN TANGAN

DENGAN ALGORITMA JARINGAN SARAF TIRUAN

MENGGUNAKAN TRANSPUTER T805

Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar sarjana pada Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya, dengan beban sebesar 6 SKS (Satuan Kredit Semester).

Tidak ada hasil karya manusia yang sempurna, demikian pula dengan buku ini. Segala saran dan kritik membangun dari semua pihak akan penulis terima dengan senang hati.

Tidak mungkin penulis dapat menyelesaikan tugas ini tanpa adanya bantuan dan dorongan dari pihak lain. Oleh karena itu penulis menyampaikan terima kasih yang sebesar-besarnya kepada :

- Bapak Ir. Soetikno, selaku Koordinator Bidang Studi Elektronika dan Koordinator Laboratorium Riset dan Pengembangan (B-203) yang telah memberikan banyak motivasi dan fasilitas selama penyelesaian tugas akhir ini.

- Bapak Ir. Harmani Suhardjo, selaku Dosen Pembimbing I yang telah membimbing penulis dalam mengerjakan tugas akhir ini.
- Bapak Ir. Hendra Kusuma, selaku Dosen Wali sekaligus Dosen Pembimbing II.
- Bapak Dr. Ir. Moch. Salehudin, M.Eng.Sc., selaku Ketua Jurusan Teknik Elektro ITS.
- Rekan-rekan di Laboratorium Bidang Studi Elektronika, terkhusus mereka yang seperjuangan di B-203.

Akhirnya semoga karya sederhana ini dapat mendatangkan manfaat bagi para pembaca dan memberikan sumbangan bagi perkembangan ilmu pengetahuan dan teknologi.

Surabaya, Agustus 1994

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Manusia selalu berusaha untuk memudahkan hidupnya dengan jalan menghindari pekerjaan-pekerjaan yang menjemukan dan memberatkan. Hal itu dilakukan dengan mengalihkan tugas-tugas tersebut kepada mesin yang salah satunya adalah komputer. Kelebihan komputer jika dibandingkan dengan manusia adalah sangat cepat dan tepat dalam melaksanakan pekerjaan atau instruksi yang sudah diformulasikan untuknya. Akan tetapi manusia jauh lebih unggul pada komputasi yang lebih kompleks, yakni menanggapi arti informasi manusiawi seperti audio dan video.

Dengan kondisi seperti ini, maka manusia terus meningkatkan kemampuan komputer agar sedekat mungkin memiliki kemampuan yang ada pada dirinya. Salah satu kemampuan yang diharapkan adalah kemampuan untuk melihat, atau jika dipersempit lagi kemampuan untuk membaca karakter atau tulisan yang ditulis oleh manusia.

Algoritma Jaringan Saraf Tiruan (Artificial Neural Network) merupakan alternatif penyelesaian yang paling tepat untuk saat ini, karena sangat luwes dan mempunyai pola pengembangan yang luas. Cara kerja jaringan ini meniru kerja dari jaringan saraf manusia yang mempunyai kemampuan beradaptasi dan menjadi tidak peka terhadap perubahan-perubahan kecil dari masukannya sampai pada

tingkat tertentu.

1.2 Permasalahan

Bila seseorang menulis dua karakter yang sama, maka bentuk kedua karakter tersebut kemungkinan besar tidak akan sama persis.Perbedaan ini akan semakin jelas bila dua karakter itu ditulis oleh orang yang berbeda. Akan tetapi tiap orang yang dapat membaca akan mengetahui karakter apa yang ditulis. Yang menjadi permasalahan adalah bagaimana mesin, dalam hal ini komputer, mentolerir perbedaan-perbedaan ini.

1.3 Pembatasan Masalah

Dengan banyaknya pola-pola karakter yang ada yang harus dikenali, maka dimensi dari jaringan akan semakin besar, dan ini jelas membutuhkan memori yang besar pula. Untuk itu dalam tugas akhir ini pengenalan dibatasi hanya pada pola angka.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah mempelajari dan mengaplikasikan jaringan saraf tiruan serta transputer dalam wujud peralatan untuk membaca angka hasil tulisan tangan dari berbagai penulis dengan berbagai variasi tulisannya.

1.5 Metodologi

Langkah-langkah yang dilakukan dalam menyelesaikan tugas akhir ini

adalah sebagai berikut :

Langkah pertama adalah studi literatur mengenai Card Transputer T805 baik secara perangkat keras maupun perangkat lunaknya, juga komunikasi dengan perangkat luar melalui serial link. Bersamaan dengan itu juga dilakukan studi literatur mengenai jaringan saraf tiruan secara umum yang kemudian lebih dikhususkan pada jaringan Error Back-Propagation.

Kemudian direncanakan dan dibuat perangkat lunak proses pelatihan dan pelaksanaan dari tugas pengenalan pola angka yang langsung dijalankan pada transputer. Untuk sementara input data disimulasikan dengan mouse.

Selanjutnya direncanakan dan dibuat perangkat keras untuk pembacaan angka. Perangkat keras dan perangkat lunak yang sudah jadi dicoba lagi dengan berbagai perubahan parameter pelatihan jaringan untuk melihat karakteristik dan performa dari jaringan setelah dihubungkan dengan perangkat keras yang merupakan bagian dari pre-processing (pengolahan awal).

Langkah terakhir adalah penyusunan naskah laporan tugas akhir.

1.6 Sistematika

Sistematika dari laporan tugas akhir ini adalah sebagai berikut :

Bab I adalah pendahuluan yang berisi mengenai latar belakang, permasalahan, pembatasan masalah, tujuan, metodologi, sistematika dan relevansi.

Bab II berisi mengenai jaringan saraf tiruan, baik pemodelan, konsep dan kaidah belajarnya, dan juga lebih dalam mengenai jaringan Error Back-Propagation. Pada bab ini juga dibahas cara kerja secara umum dari jaringan saraf

yang sesungguhnya.

Bab III berisi mengenai uraian prosesor yang digunakan yakni transputer T805. Dijelaskan karakteristik dari transputer secara umum dan juga secara khusus pada type T805.

Bab IV berisi perencanaan sistim perangkat keras, yakni peralatan pengambil data pola angka atau scanner.

Bab V berisi perencanaan perangkat lunak, baik perangkat lunak untuk pemrosesan awal maupun Jaringan Saraf Tiruan.

Bab VI berisi mengenai proses pengujian sistem yang telah dibuat.

Bab VII adalah penutup yang berisi kesimpulan dan saran.

1.7. Relevansi

Hasil Tugas akhir ini diharapkan dapat menjadi tambahan pengetahuan mengenai transputer dan jaringan saraf tiruan bagi mahasiswa bidang studi Elektronika khususnya dan bagi mahasiswa Teknik Elektro pada umumnya. Dan juga diharapkan menjadi pemicu bagi riset-riset yang lebih jauh dalam bidang ini.

BAB II

JARINGAN SARAF TIRUAN

2.1 Pendahuluan

Penemuan mesin-mesin sederhana, seperti tuas, roda dan katrol, sampai mesin-mesin canggih semuanya ditujukan menggantikan manusia dalam tugas-tugas yang sulit dan menjemukan. Sistem saraf tiruan merupakan salah satu mesin yang mempunyai potensi kuat untuk lebih meningkatkan kualitas hidup manusia.

Kemampuan komputasi dari jaringan saraf tiruan didasarkan pada harapan bahwa dapat disimulasikannya beberapa fleksibilitas dan kemampuan otak manusia. Jaringan ini terdiri dari banyak sel yang dikenal sebagai *neuron*. Masing-masing neuron terhubung dengan yang lainnya dengan aturan tertentu dan dengan kekuatan hubung yang disebut bobot (*weight*). Sel-sel ini bertugas melakukan perhitungan secara paralel terdistribusi dan juga sebagai unit *threshold* yang akan mengaktifkan outputnya bila total dari semua inputnya melewati level tertentu.

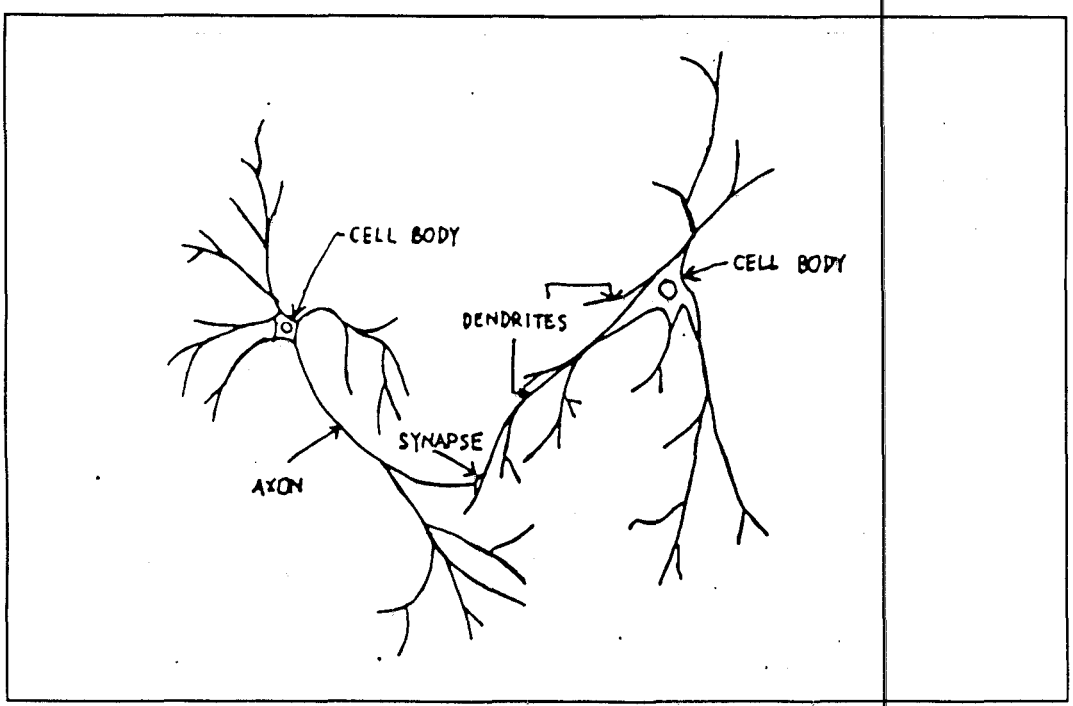
Berbeda dengan komputer konvensional yang diprogram untuk suatu tugas yang telah ditentukan, jaringan saraf tiruan harus belajar atau dilatih. Semakin banyak yang dilatihkan padanya, jaringan akan semakin pandai. Dalam perangkat lunak untuk implementasi jaringan saraf tiruan, yang dibuat adalah program pelatihannya, bukan apa yang harus dilakukannya. Jaringan saraf tiruan akan menemukan sendiri fungsi transfer antara input dan output.

Ada baiknya sebelum beranjak lebih jauh ke jaringan saraf tiruan untuk meninjau sepintas mengenai kerja dari sel saraf yang sesungguhnya.

2.2 Sel Saraf Biologis

2.2.1 Bagian-bagian sel saraf biologis

Neuron adalah elemen dasar dari sistem saraf biologis. Ini merupakan sel yang sama dengan semua sel tubuh, tetapi spesialisasi tertentu akan membentuk fungsi komputasi dan komunikasi tertentu dalam otak.



Gambar 2-1
Bagian-bagian dari neuron[1]

[1]Wasserman, Philip D., *Neural Network Theory and Practice*, Van Nostrand Reinhold, New York, 1989, hal, 192.

Seperti terlihat pada Gambar 2-1, neuron terdiri dari tiga bagian utama : badan sel (soma), akson dan dendrit. Dendrit menerima sinyal-sinyal dari sel lain pada titik hubung yang disebut sinapsis. Dari sana sinyal-sinyal tersebut dilewatkan badan sel yang kemudian dirata-rata dengan sinyal-sinyal lain yang sejenis. Jika hasil rata-rata ini, selama interval waktu yang pendek, cukup besar, maka sel akan membuat outputnya aktif dan menghasilkan pulsa pada akson. Akson meneruskan sinyal ini ke sel yang berikutnya.

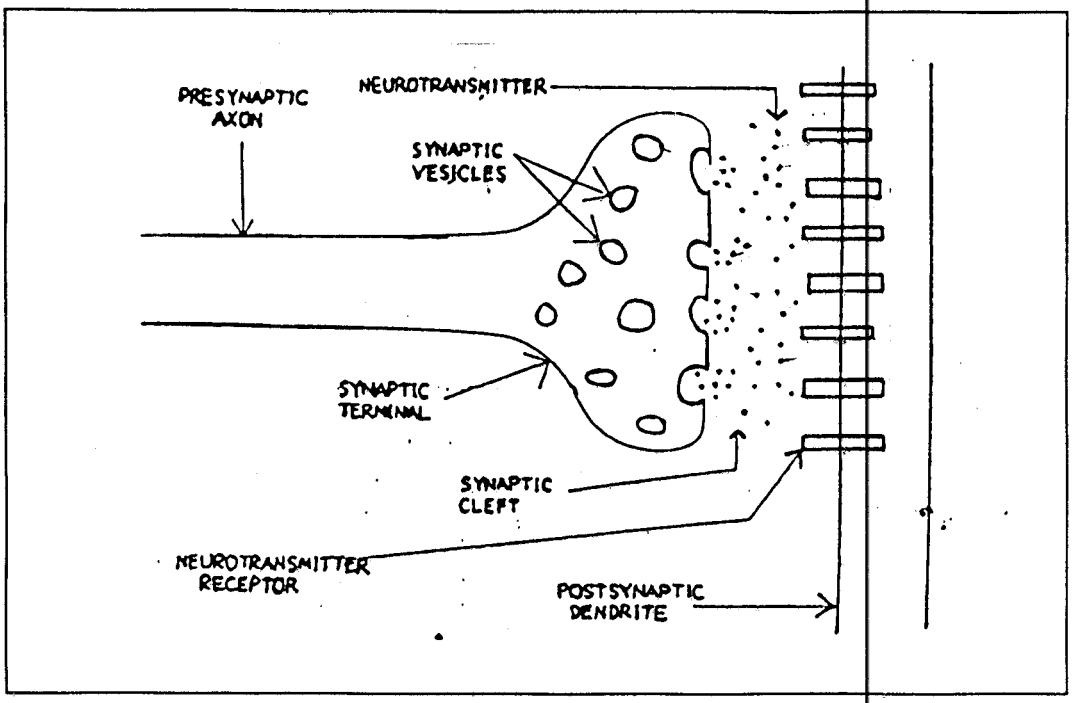
2.2.1.1 Badan Sel

Neuron dari otak orang dewasa tidak melakukan regenerasi. Ini berarti semua komponennya harus diganti secara kontinyu dan dibutuhkan pembaharuan bahan. Sebagian besar aktifitas pemeliharaan ini dilakukan dalam badan sel. Badan sel juga mengatur energi dari neuron dan aktifitas-aktifitas lain dalam sel.

2.2.1.2 Dendrit

Sinyal-sinyal input yang berasal dari neuron-neuron yang lain memasuki sel melalui dendrit. Pada dendrit ini terdapat hubungan sinaptik, dimana sinyal-sinyal diterima dari akson yang lain. Tidak seperti rangkaian listrik, biasanya tidak ada hubungan fisik ataupun elektrik pada sinapsis. Namun terdapat celah sempit yang dikenal dengan celah sinaptik *(synaptic cleft)* yang memisahkan dendrit dari akson. Zat-zat kimia tertentu dilepas oleh akson ke dalam celah sinaptik dan disemburkan ke dendrit. Zat-zat kimia ini, yang dikenal sebagai *neurotransmitter*, dilewatkan ke reseptor pada dendrit dan masuk ke badan sel.

Badan sel menggabungkan sinyal-sinyal yang diterima melalui semua dendritnya dan jika sinyal resultannya di atas nilai ambangnya akan dihasilkan sebuah pulsa yang dipancarkan melalui akson ke neuron yang lain.



Gambar 2-2
Sinapsis[2]

2.2.1.3 Akson

Tidak seperti dendrit, akson aktif secara elektris dan bekerja sebagai kanal output dari neuron. Akson adalah piranti threshold non-linear yang menghasilkan pulsa tegangan yang disebut dangan potensial aksi (*action potensial*). Potensial aksi yang mempunyai durasi seekitar 1 ms ini, akan terjadi bila potensial di dalam soma melebihi ambang batas tertentu.

Berdekatan dengan ujungnya, akson mempunyai banyak cabang, yang

---

[2]ibid., hal. 195.

masing-masing berakhir pada sinapsis, dimana sinyal dipancarkan ke neuron yang lain. Pada koneksi sinaptik terdapat struktur-struktur bola yang dikenal sebagai gelembung sinaptik *(synaptic vesicle)*. Gelembung-gelembung ini berisi sejumlah besar molekul-molekul *neurotransmitter*. Jika potensial aksi saraf melewati akson, beberapa dari gelembung ini akan melepas isinya ke celah sinapsis. Diperlukan lebih dari satu potensial aksi sebelum sinapsis ter-trigger. Neurotransmitter yang dilepas oleh sinapsis akan terdifusi ke dalam celah, dan secara kimia akan mengaktifkan gerbang-gerbang dendrit pada dendrit, dimana jika gerbang ini terbuka akan membuat ion-ion bermuatan mengalir. Aliran ion inilah yang menggantikan potensial dendrit, dan memberikan pulsa tegangan pada dendrit, yang kemudian diteruskan ke badan sel. Masing-masing dendrit dapat mempunyai banyak akson yang beraksi padanya, sehingga diperoleh interkoneksi yang padat. Pada hubungan sinapsis, jumlah dari gerbang yang terbuka pada dendrit tergantung pada jumlah neurotransmitter yang dilepas. Beberapa sinapsis merangsang dendrit yang mereka pengaruhi, sementara yang lainnya menghalangi. Ini bersesuaian dengan potensial lokal dari dendrit dalam arah positip dan negatip.

2.2.2 Proses belajar sistem biologis

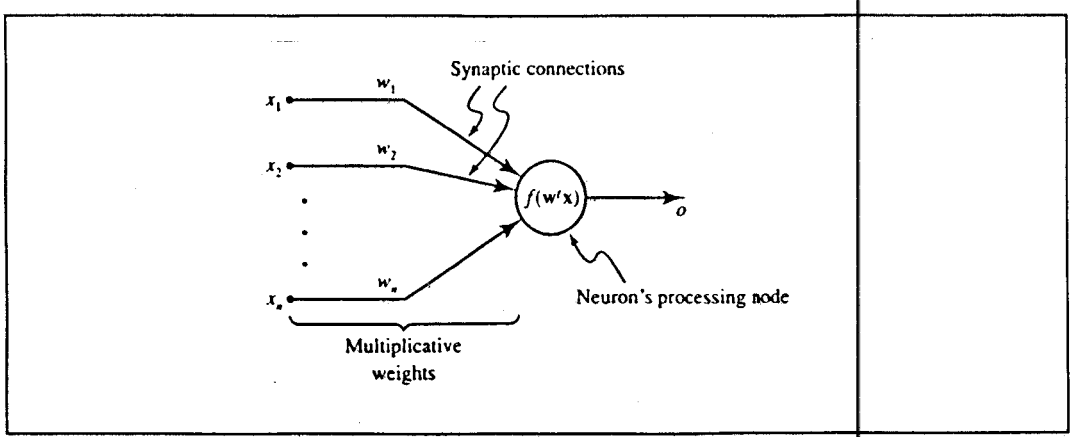
Suatu sistem sel saraf biologis dikatakan belajar jika terjadi modifikasi hubungan efektif antara satu sel dengan sel yang lain, pada hubungan sinapsis. Mekanisme untuk memperolehnya adalah dengan memudahkan pelepasan lebih banyak neurotransmitter. Ini mempunyai efek membuka lebih banyak gerbang pada dendrit pada sisi post-sinapsis dari hubungan sinapsis. Pengaturan kopling

sehingga memperkuat koneksi yang baik adalah hal yang penting bagi model jaringan saraf tiruan. Kopling efektif ini dikenal dengan pembobotan.

2.3 Pemodelan Jaringan Saraf

2.3.1 Pemodelan Sel Saraf

Fungsi dasar dari sebuah neuron adalah menjumlahkan semua inputnya, memberikan sebuah output atau mengaktifkan outputnya bila jumlah ini melebihi nilai tertentu yang disebut nilai batas *(threshold)*. Gambar 2-3 memperlihatkan model dari neuron.



Gambar 2-3
Model Neuron[3]

Output dari neuron dinyatakan sebagai berikut :

$$o = f(w^t x) \tag{2-1a}$$

[3]Zurada, Jacek M., *Introducction to Artificial Neural Systems*, Info Access Distribution Pte Ltd., Singapore, 1992, hal. 32.

$$o = f\left(\sum_{i=1}^{n} w_i x_i\right) \qquad (2\text{-}1b)$$

dimana **w** adalah vektor bobot yang didefinisikan sebagai

$$w \triangleq [w_1 \; w_2 \; \ldots \; w_n]^t$$

dan **x** adalah vektor input :

$$x = [x_1 \; x_2 \; \ldots \; x_n]^t$$

Vektor-vektor di atas adalah vektor kolom, tanda *t* menunjukkan transpose vektor.

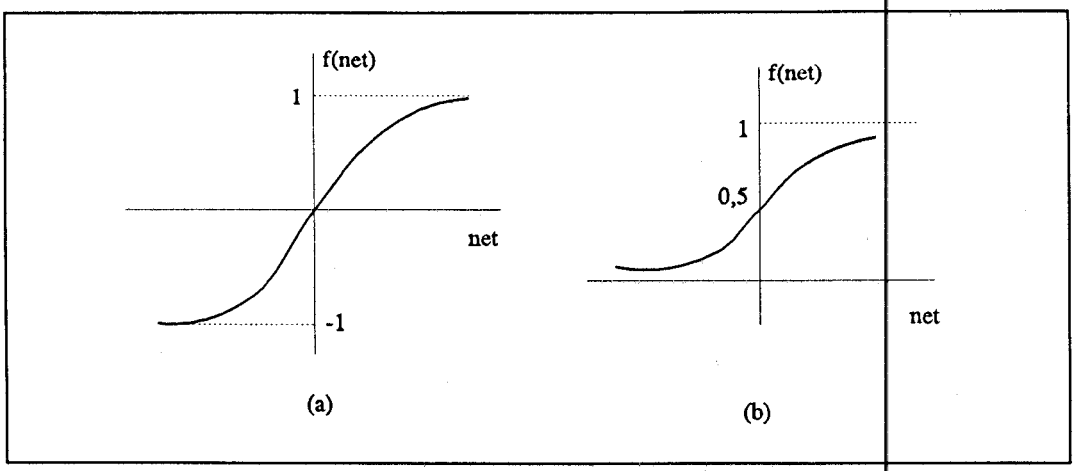
Fungsi $f(w^t x)$ dikenal sebagai *fungsi aktivasi*. Daerah asal fungsi ini adalah himpunan nilai aktivasi, *net*, dari neuron buatan, sehingga sering ditulis sebagai *f(net)*. Variabel *net* didefinisikan sebagai perkalian skalar dari vektor bobot dan vektor input :

$$net \triangleq w^t x \qquad (2\text{-}2)$$

Ada dua fungsi aktivasi yang biasa digunakan, yaitu *bipolar* dan *unipolar*. Fungsi aktivasi bipolar dinyatakan sebagai berikut :

$$f(net) \triangleq \frac{2}{1 + e^{-\lambda net}} - 1 \qquad (2\text{-}3a)$$

dimana λ (>0) proporsional dengan penguatan neuron menentukan kecuraman dari kurva fungsi. Persamaan (2-3a) merupakan fungsi aktivasi kontinyu. Untuk $\lambda \rightarrow \infty$ fungsi aktivasi kontinyu menjadi fungsi aktivasi biner (Persamaan (2-3b)).



Gambar 2-4
Fungsi aktivasi kontinyu : (a) bipolar dan (b) unipolar

$$f(net) \triangleq sgn(net) = \begin{cases} +1, & net > 0 \\ -1, & net < 0 \end{cases} \qquad (2\text{-}3b)$$

Dengan menggeser dan men-skala fungsi bipolar akan diperoleh fungsi aktivasi unipolar kontinyu dan biner sebagai berikut :

$$f(net) \triangleq \frac{1}{1+e^{-\lambda net}} \qquad (2\text{-}4a)$$

$$f(net) \triangleq \begin{cases} 1, & net > 0 \\ 0, & net < 0 \end{cases} \qquad (2\text{-}4b)$$

Fungsi aktivasi dapat dianggap seperti pendefinisian penguatan tak-linier dalam sistem elektronik analog. Penguatannya dihitung dengan cara mencari rasio perubahan pada *o* terhadap perubahan kecil pada *net*. Jadi, penguatan merupakan kemiringan kurva pada suatu tingkat eksitasi tertentu, dan nilainya berubah dari harga yang kecil pada eksitasi negatif yang besar menjadi harga yang besar pada eksitasi nol, dan nilainya kembali mengecil seiring dengan eksitasi yang semakin besar dan positip.

Model sederhana neuron buatan ini mengabaikan banyak sekali karakteristik neuron biologis yang sebenarnya. Misalnya, model ini tidak memperhitungkan waktu tunda yang mempengaruhi dinamika sistem; input dari neuron buatan ini segera menghasilkan output. Lebih jauh lagi, model ini tidak melibatkan fungsi modulasi frekuensi yang oleh periset dianggap penting. Meskipun demikian, jaringan saraf buatan ini menunjukkan hasil yang cukup baik.

2.3.2 Model-model Jaringan Saraf Tiruan

Jaringan saraf tiruan dapat didefinisikan sebagai interkoneksi neuron-neuron, seperti didefinisikan pada persamaan (2-1) sampai (2-4), sedemikian hingga output neuron-neuron itu terhubung, melalui bobot-bobot, ke neuron-neuron lain. Seiring dengan banyaknya riset yang dilakukan dalam bidang ini, definisi-definisi baru telah dikembangkan. Tetapi pada dasarnya ada dua model dasar dari jaringan saraf tiruan, jaringan umpan-maju *(feedforward network)* dan jaringan umpan-balik *(backward network)*.

2.3.2.1 Jaringan Umpan-maju

Gambar 2-5 memperlihatkan arsitektur jaringan saraf tiruan umpan-maju. Jaringan ini mempunyai m sel yang menerima n input. Masing-masing vektor output dan inputnya yaitu :

$$o = [o_1 \; o_2 \; \ldots \; o_m]^t$$
$$x = [x_1 \; x_2 \; \ldots \; x_n]^t \qquad (2\text{-}5)$$

Bobot w_{ij} menghubungkan sel neuron ke-i dengan input sel ke-j. Sehingga nilai

aktivasi dari sel ke-*i* dapat dituliskan sebagai berikut :

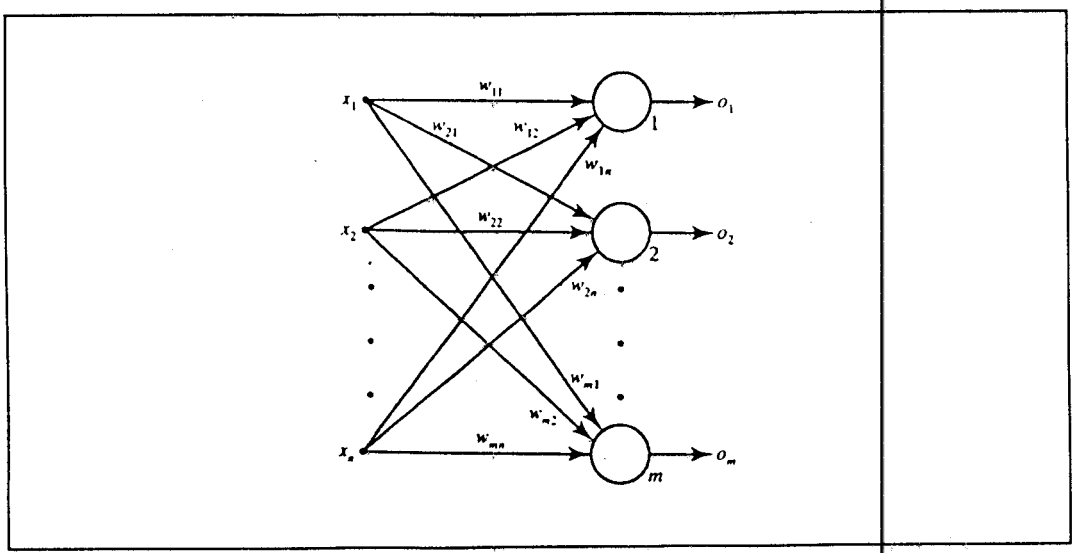
$$net_i = \sum_{j=i}^{n} w_{ij} x_j, \qquad untuk \quad i = 1,2,\ldots,m \tag{2-6}$$

Kemudian dengan menggunakan fungsi aktivasi $f(net_i)$, masing-masing sel akan menghasilkan output :

$$o_i = f(w_i^t x), \qquad untuk \; i = 1,2,\ldots,m \tag{2-7}$$

dimana vektor bobot w_i berisi bobot yang langsung terhubung dengan sel ke-*i*, dan didefinisikan sebagai :

$$w_i \triangleq [w_{i1} \; w_{i2} \; \ldots \; w_{i\in}] \tag{2-8}$$



Gambar 2-5
Jaringan umpan-maju lapisan tunggal[4]

Dengan menggunakan operator matriks Γ, pemetaan input x ke output o dapat dinyatakan :

---

[4]ibid., hal. 38.

$$o = \Gamma[Wx] \tag{2-9}$$

Pemetaan jenis umpan-maju ini merupakan pemetaan langsung *(instant)*, karena tidak adanya delay antara input **x** dan output **o**. Dapat ditulis lagi dengan penunjuk waktu t sebagai :

$$o(t) = \Gamma[Wx(t)] \tag{2-10}$$

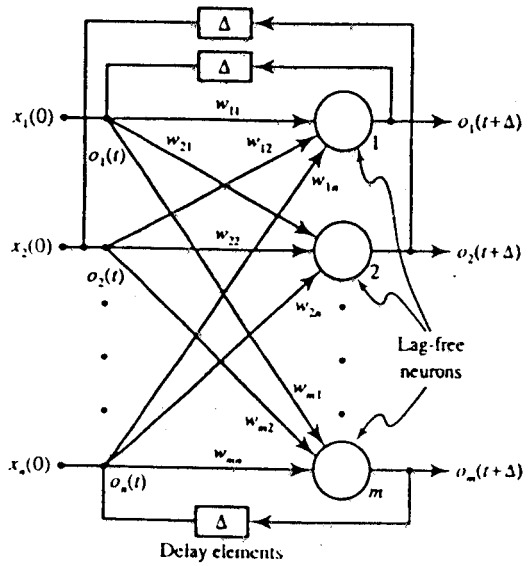
Tipe jaringan ini dapat disusun secara paralel untuk menghasilkan jaringan lapisan jamak *(multilayer network)*. Dalam jaringan yang demikian, output dari satu lapisan merupakan input dari lapisan berikutnya.

2.3.2.2 Jaringan Umpan-balik

Jaringan umpan-balik dapat diperoleh dari jaringan umpan-maju dengan jalan menghubungkan output dari sel ke inputnya.

Inti dari jaringan umpan-balik ini adalah untuk dapat mengontrol output o_i melalui output o_j, untuk $j = 1, 2, \ldots, m$. Kontrol ini sangat berarti jika output saat ini, $\mathbf{o}(t)$, mongontrol output yang berikutnya, $\mathbf{o}(t+\Delta)$. Selisih waktu Δ antara t dan $t+\Delta$ adalah elemen delay pada loop umpan-balik (Gambar 2-6). Pemetaan $\mathbf{o}(t)$ ke $\mathbf{o}(t+\Delta)$ dapat ditulis sebagai berikut :

$$O(t+\Delta) = \Gamma[Wx(t)] \tag{2-11}$$



Gambar 2-6
Jaringan umpan-balik[5]

Yang penting dalam jaringan umpan-balik yaitu input $\mathbf{x}(t)$ hanya digunakan untuk inisialisasi jaringan sehingga $\mathbf{o}(0) = \mathbf{x}(0)$. Setelah itu, untuk $t > 0$, input ditiadakan dan sistem akan berjalan sendiri.

2.4 Pemrosesan Neural

Jaringan saraf tiruan menghitung outputnya (o) untuk suatu input (x) yang diberikan. Proses perhitungan ini dikenal dengan sebutan *recall*. Recall merupakan fase pemrosesan yang sesungguhnya untuk jaringan saraf tiruan dan tujuannya adalah untuk mengambil kembali informasi yang disimpan di dalamnya. Recall bersesuaian dengan men-dekode informasi yang telah dienkode sebelumnya di

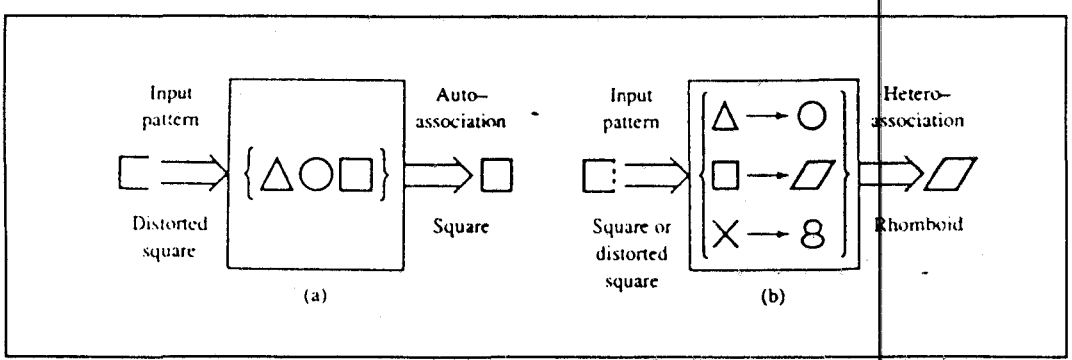
[5]ibid., hal. 42.

dalam jaringan. Ada dua proses recall dasar pada jaringan saraf tiruan, yakni: Asosiasi dan Klasifikasi.

2.4.1 Asosiasi

Misalkan ada sekelompok pola disimpan dalam jaringan. Kemudian jika jaringan diberi input sebuah pola yang sama dengan salah satu anggota kelompok tersebut. Maka jaringan akan mengasosiasikan input tersebut dengan pola yang terdekat. Proses ini disebut asosiasi. Biasanya pola input yang kurang lengkap atau kurang sempurna memberikan semacam isyarat atau pola dasar untuk pengambilan bentuk aslinya. Gambar 2-7(a) mengilustrasikan proses ini.

Asosiasi dari pola-pola input dapat juga disimpan dalam heteroasosiasi. Dalam proses heteroasosiatif, disimpan asosiasi antara pasangan-pasangan pola.



Gambar 2-7
Respon Asosiasi : (a) autoasosiasi dan (b) heteroasosiasi[6]

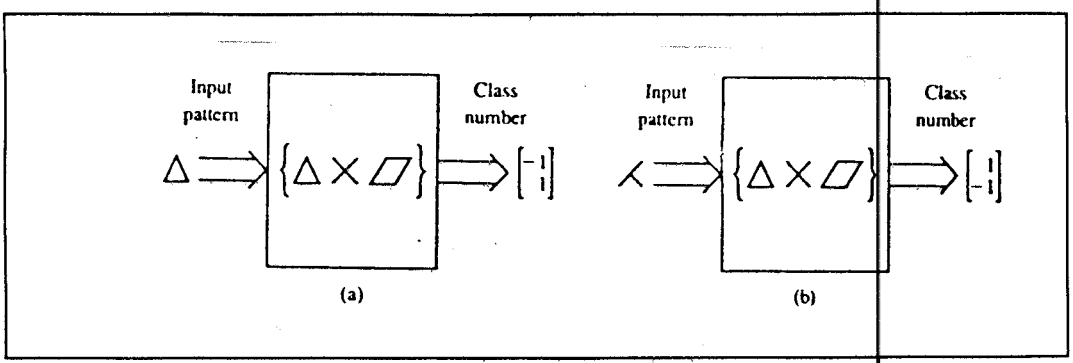
Gambar 2-7(b) mengilustrasikan proses ini. Pola input bujur sangkar terdistorsi yang diberikan pada input menghasilkan jajaran genjang pada output. Dapat diartikan bahwa jajaran genjang dan bujur sangkar merupakan satu pasangan dari

[6]ibid., hal. 53.

pola-pola yang disimpan.

2.4.2 Klasifikasi

Misalkan sekelompok pola input dibagi dalam sejumlah kelas atau katagori. Dalam me-respon ke salah satu pola input dari kelompok tersebut, klasifier me-recall informasi dengan memperhatikan keanggotaan kelas dari pola input. Biasanya, kelas-kelas dinyatakan dengan vektor output dengan nilai diskrit. Gambar 2-8(a) memperlihatkan respon klasifikasi untuk pola-pola yang dimiliki oleh tiga kelas.



Gambar 2-8
Respon Klasifikasi : (a) klasifikasi dan (b) recognition[7]

Klasifikasi dapat dimengerti sebagai salah satu kasus dari heteroasosiasi. Asosiasi antara pola input dan anggota kedua dari pasangan heteroasosiatif, yang mengindikasikan nomor kelasnya. Jika respon jaringan yang diinginkan adalah nomor kelas tetapi pola inputnya tidak sama persis dengan salah satu pola yang ada, prosesnya dikenal sebagai *recognition*. Proses recognition diperlihatkan pada Gambar 2-8(b).

---

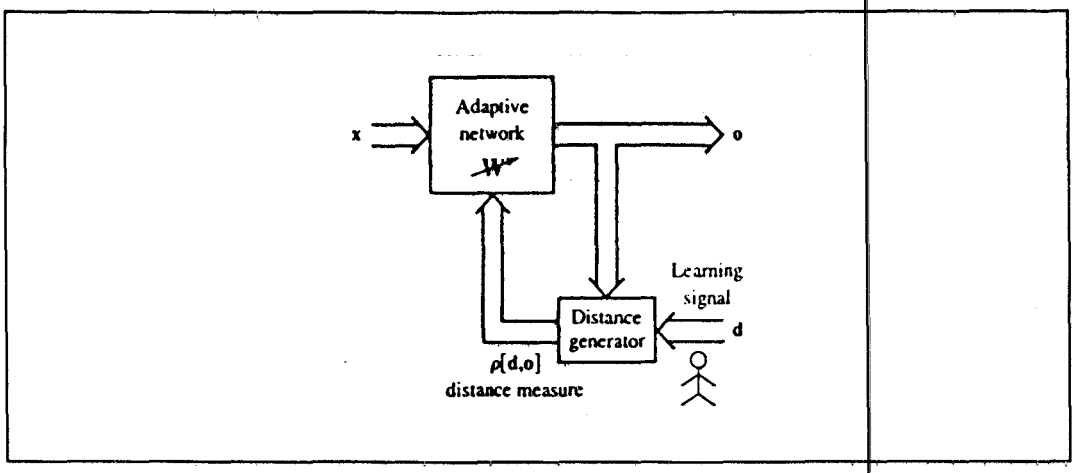
[7]ibid., hal. 54.

2.5 Konsep Belajar

Karakteristik yang sangat menarik dari jaringan saraf tiruan adalah kemampuannya untuk belajar. Setelah tahap belajar selesai, jaringan diharapkan mampu untuk menghasilkan serangkaian output yang diinginkan jika jaringan diberi serangkaian input. Setiap input atau output teracu sebagai vektor. Proses belajar atau pelatihan dari jaringan ini dilakukan dengan jalan membangkitkan vektor input secara berurutan dan mengatur bobot dengan prosedur tertentu. Selama proses belajar, bobot secara bertahap akan berubah menuju keadaan yang sedemikian rupa sehingga setiap vektor input akan menghasilkan vektor output yang diharapkan.

2.5.1 Belajar dengan Pengawasan (Supervised learning)

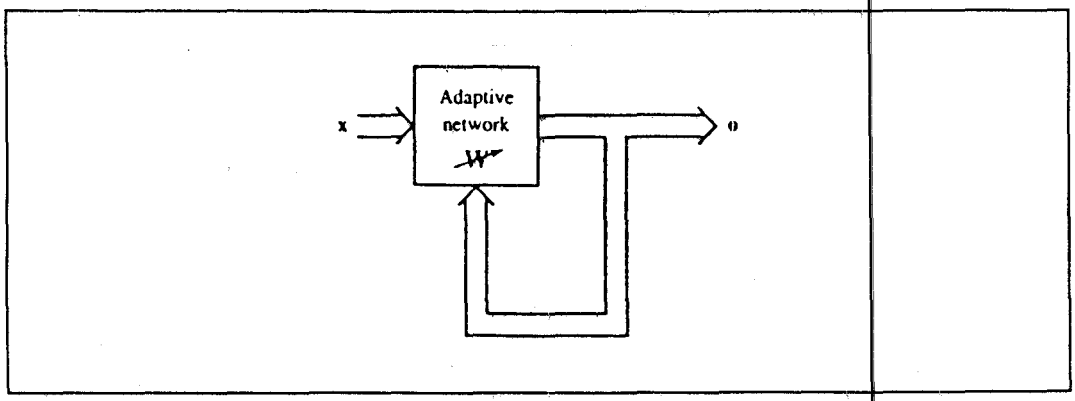
Pada belajar dengan pengawasan dibutuhkan adanya vektor lain, yaitu vektor target (**d**). Vektor target berisi output yang diinginkan dari vektor input (**x**) yang diberikan. Sebuah vektor masukan dibangkitkan, output dari jaringan dihitung dan hasilnya dibandingkan dengan vektor target yang bersesuaian. Selisih antara output dengan target, rho[**d**,**o**], diumpanbalikkan. Selanjutnya matriks bobot **W** diatur menurut algoritma yang akan meminimkan selisih tersebut.



Gambar 2-9
Belajar dengan pengawasan[8]

2.5.2 Belajar tanpa Pengawasan (Unsupervised learning)

Belajar tanpa pengawasan tidak membutuhkan adanya vektor target, oleh karenanya tidak ada langkah membandingkan dengan output ideal yang diinginkan. Karena output yang diinginkan tidak diketahui, informasi kesalahan eksplisit tidak dapat digunakan untuk memperbaiki respon jaringan.



Gambar 2-10
Belajar tanpa pengawasan[9]

[8]ibid., hal. 57.

[9]loc.cit.

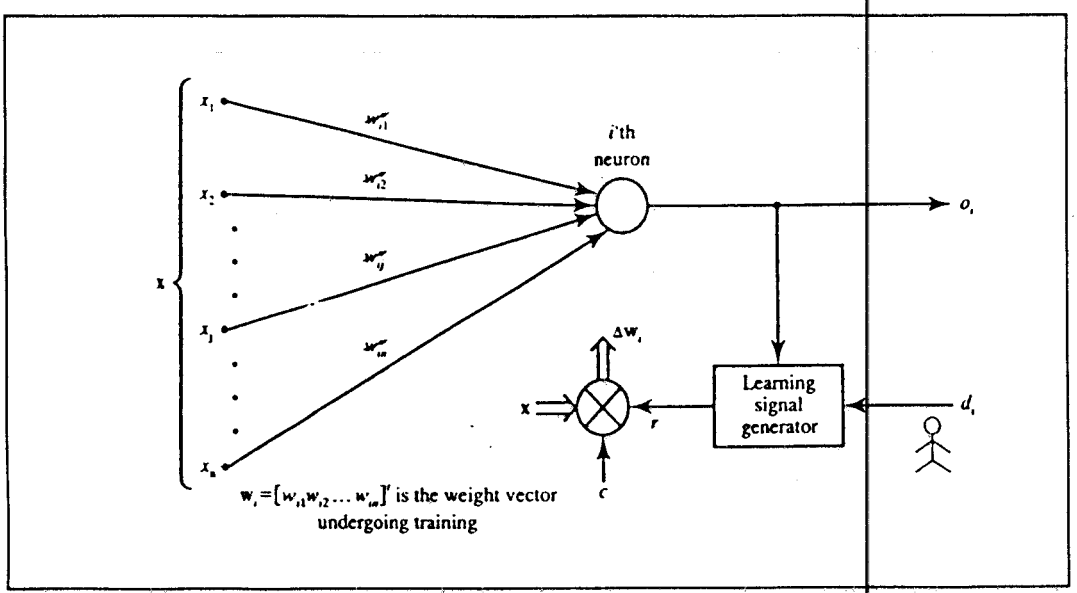
Rangkaian pelatihan hanya berisi vektor input (x) saja. Algoritma belajar mengubah bobot jaringan untuk menghasilkan vektor output yang sesuai.

2.6 Kaidah Belajar

2.6.1 Kaidah belajar umum

Ada berbagai kaidah belajar bagi jaringan saraf tiruan, tergantung pada jaringan yang diaplikasikan. Tetapi semua kaidah belajar ini mempunyai kaidah umum.

Gambar 2-11 memperlihatkan jaringan saraf tiruan yang akan dilatih (d_i hanya berlaku untuk mode belajar dengan pengawasan). Vektor $\mathbf{w}_i$ yang mempunyai komponen w_{ij} menghubungkan input ke-*j* dengan neuron ke-*i*. Secara umum input ke-*j* dapat berupa output dari neuron lain atau input eksternal.



Gambar 2-11
Neuron yang dilatih[10]

[10]ibid., hal. 60.

Aturan belajar umum dapat dinyatakan sebagai berikut : Vektor bobot $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]$ bertambah secara proporsional dengan hasil perkalian vektor input x dengan sinyal belajar r. Sinyal belajar r adalah fungsi dari $w_i x$ -dan sinyal d_i (untuk mode dengan pengawasan). Jadi untuk Gambar 2-10 dapat dinyatakan :

$$r = r(w_i, x, d_i) \tag{2-12}$$

Kenaikan vektor bobot w_i yang diperoleh dari step belajar pada saat t adalah :

$$\Delta w_i(t) = cr[w_i(t), x(t), d_i(t)]x(t) \tag{2-13}$$

dimana c adalah bilangan positif yang disebut konstanta belajar (*learning constant*) yang menentukan laju belajar. Vektor bobot yang diadaptasi pada saat *t* dipakai pada siklus belajar yang berikutnya :

$$w_i(t+1) = w_i(t) + cr[w_i(t), x(t), d_i(t)]x(t) \tag{2-14a}$$

Untuk selanjutnya persamaan di atas akan ditulis dengan memakai superscript untuk indeks dari step latihan waktu-diskrit. Untuk step ke-k ditulis sebagai :

$$w_i^{k+1} = w_i^k + cr(w_i^k, x^k, d_i^k)x^k \tag{2-14b}$$

Untuk belajar dengan waktu-kontinyu persamaan (2-14) dapat dinyatakan dengan :

$$\frac{dw_i(t)}{dt} = crx(t) \tag{2-15}$$

2.6.2 Kaidah belajar Perseptron

Dalam kaidah belajar perseptron, sinyal belajar adalah selisih antara respon

yang diinginkan dengan respon sesungguhnya. Metode belajarnya dengan pengawasan dan sinyal belajarnya didefinisikan sebagai :

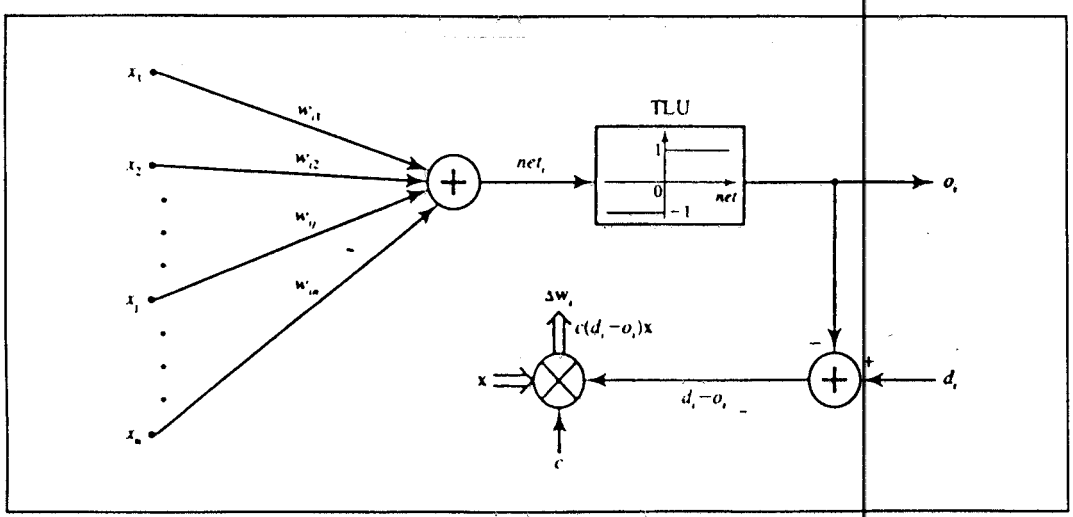
$$r \triangleq d_i - o_i \tag{2-16}$$

dimana $o_i = \text{sgn}(\mathbf{w}_i^t\mathbf{x})$ dan d_i adalah adalah respon yang diinginkan. Gambar 2-12 memperlihatkan proses belajar kaidah perceptron.

Pengaturan bobot dalam kaidah ini, $\Delta\mathbf{w}_i$ dan Δw_{ij}, diperoleh sebagai berikut :

$$\Delta w_i = c[d_i - sgn(w_i^t x)]x \tag{2-17a}$$

$$\Delta w_{ij} = c[d_i - sgn(w_i^t x)]x_j, \qquad untuk \quad j = 1,2,\ldots,n \tag{2-17b}$$



Gambar 2-12
Kaidah belajar Perseptron[11]

Kaidah belajar ini hanya berlaku untuk respon neuron biner. Menurut kaidah ini, bobot-bobot diatur hanya jika o_i tidak benar. Dan karena respon yang diinginkan

[11]ibid., hal. 64.

hanya dua macam yakni 1 atau -1, maka pengaturan bobot dirumuskan lagi sebagai:

$$\Delta w_i = \pm 2cx \tag{2-18}$$

dimana tanda plus dipakai bila $d_i = 1$, dan $\text{sgn}(\mathbf{w}^t\mathbf{x}) = -1$, dan tanda minus dipakai bila $d_i = -1$, dan $\text{sgn}(\mathbf{w}^t\mathbf{x}) = 1$.

2.6.3 Kaidah belajar Delta

Kaidah belajar delta hanya berlaku untuk fungsi aktivasi kontinyu, dan dalam mode belajar dengan pengawasan (*supervised*). Sinyal belajar untuk kaidah ini disebut delta dan didefinisikan sebagai berikut :

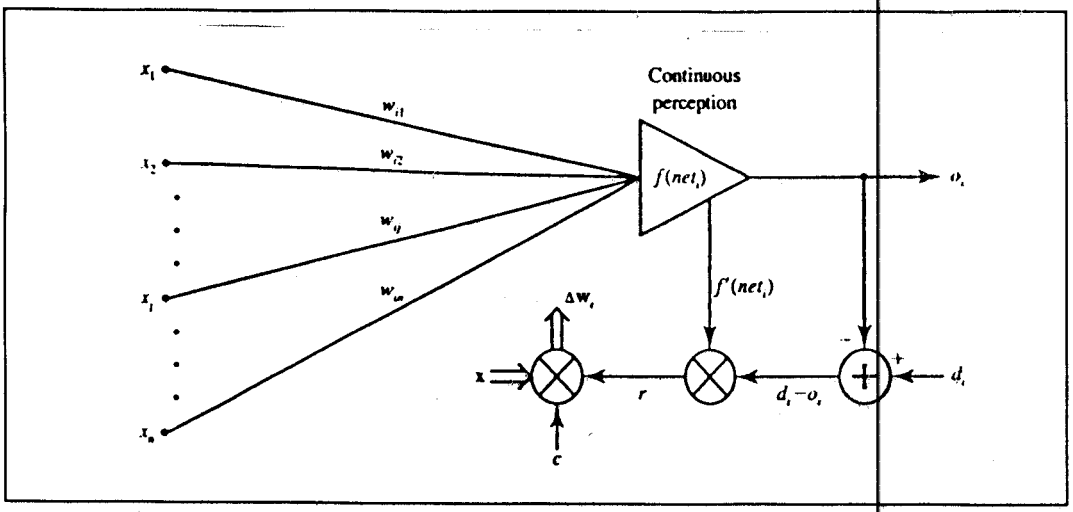
$$r \triangleq [d_i - f(w_i^t x)]\, f'(w_i^t x) \tag{2-19}$$

Faktor $f'(w_i^t x)$ adalah turunan dari fungsi aktivasi yang dihitung untuk $net = w_i^t x$. Gambar 2-13 memperlihatkan kaidah belajar delta. Aturan belajar ini diturunkan dari kondisi *least square error* antara o_i (output) dan d_i (target). Error didefinisikan sebagai :

$$E \triangleq \frac{1}{2}(d_i - o_i)^2 \tag{2-20a}$$

yang ekivalen dengan :

$$E = \frac{1}{2}[d_i - f'(w_i^t x)]^2 \tag{2-20b}$$



Gambar 2-13
Kaidah belajar delta[12]

Diperoleh nilai vektor gradien error sebagai berikut :

$$\nabla E = -(d_i - o_i) f'(w_i^t x) x \qquad (2\text{-}21a)$$

Komponen-komponen dari vektor gradien yaitu :

$$\frac{\partial E}{\partial w_{ij}} = -(d_i - o_i)\, f'(w_i^t x) x_j \qquad , \textit{untuk} \quad j = 1,2,\ldots,n \qquad (2\text{-}21b)$$

Karena untuk minimisasi error dibutuhkan perubahan bobot dalam arah gradien negatif, maka diambil :

$$\Delta w_i = -\eta \nabla E \qquad (2\text{-}22a)$$

dimana η adalah konstanta positif. Dari persamaan (2-20) dan (2-21) diperoleh :

$$\Delta w_i = \eta (d_i - o_i)\, f'(net_i) x \qquad (2\text{-}22b)$$

Atau untuk pengaturan salah satu bobot manjadi :

---

[12]ibid., hal. 67.

$$\Delta w_{ij} = \eta(d_i - o_i)\, f'(net_i) x_j \qquad (2\text{-}22c)$$

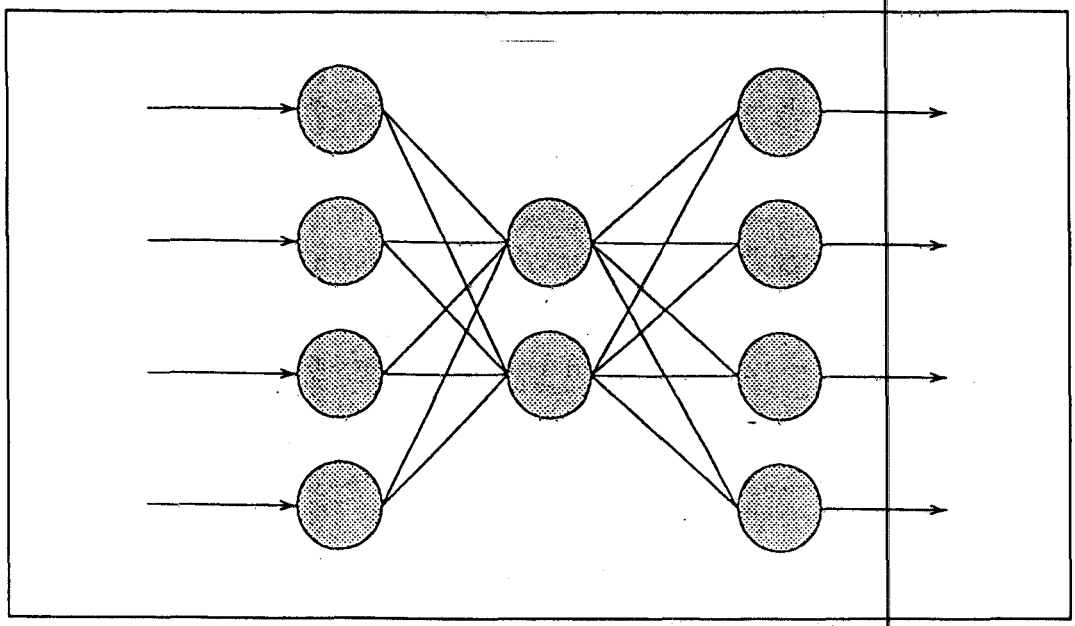
2.7 Jaringan Error Back-Propagation

2.7.1 Pandangan Umum

Error Back-Propagation adalah jaringan saraf tiruan yang paling banyak digunakan dan telah berhasil diterapkan dalam berbagai bidang. Jika pada jaringan ini diberikan suatu pola input, maka akan memberikan pola output yang bersesuaian. Dan di antara berbagai macam model jaringan saraf tiruan, Error Back-Propagation adalah model yang paling mudah dimengerti. Prosedur belajarnya didasarkan pada konsep yang relatif sederhana: jika jaringan memberikan jawaban yang salah, bobot-bobotnya akan dikoreksi sedemikian hingga error berkurang dan sebagai hasilnya respon jaringan kemudian akan lebih mendekati kebenaran.

Error Back-Propagation mempunyai paling sedikit dua lapisan, yakni lapisan input *(input layer)* dan lapisan output *(output layer)*. Jika ditambahkan satu lapisan atau lebih di antara lapisan input dan lapisan output, maka lapisan tambahan ini dinamakan lapisan tengah *(hidden layer)*. Gambar 2-14 memperlihatkan jaringan Error Back-Propagation dengan tiga lapisan.

Error Back-Propagation memakai kaidah belajar delta. Jika jaringan ini tidak mempunyai lapisan tengah, untuk prosedur penghitungan pengaturan bobot dipakai kaidah belajar delta biasa. Sedangkan untuk menghitung pengaturan bobot untuk lapisan tengah digunakan kaidah belajar delta yang sudah digeneralisasi.



Gambar 2-14
Jaringan Error Back-Propagation tiga lapisan[13]

2.7.2 Kaidah Belajar Delta untuk Lapisan Perseptron Jamak

Gambar 2-15 memperlihatkan jaringan lapisan tunggal dengan perseptron kontinyu. Nilai input dilambangkan dengan y_j dan output dengan o_k. Maka y_j, untuk $j = 1, 2, \ldots, J$ dan o_k untuk $k = 1, 2, \ldots, K$ masing-masing adalah nilai sinyal pada kolom sel ke-j dan kolom sel ke-k. Bobot w_{kj} menghubungkan output dari sel ke-j dan input dari sel ke-k.

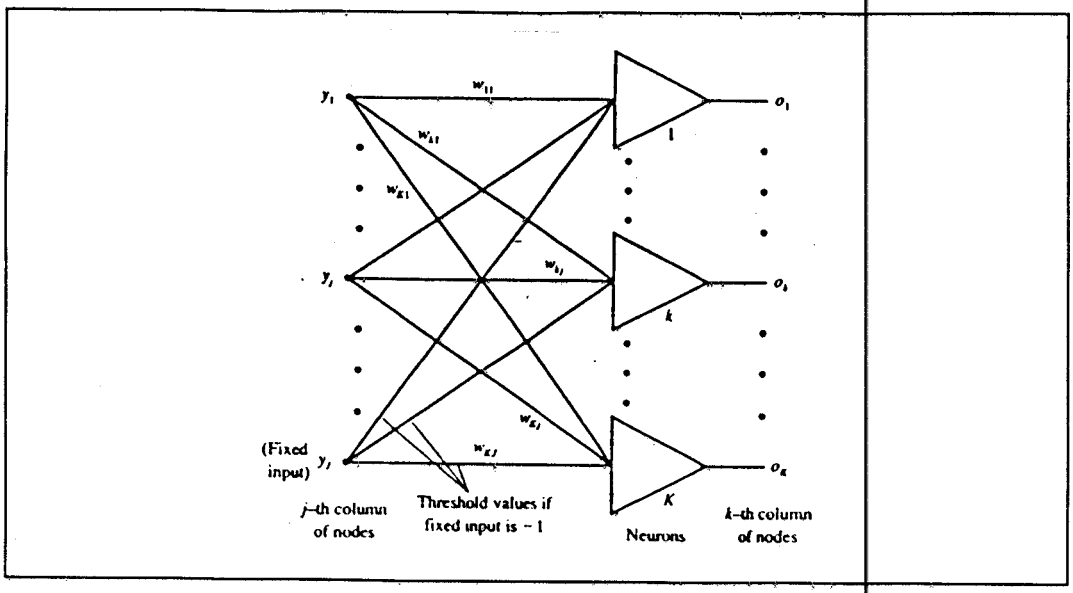
Pernyataan error pada persamaan (2-20a) sekarang digeneralisasi untuk semua error pada output $k = 1, 2, \ldots, K$:

$$E_p = \frac{1}{2}\sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \qquad (2\text{-}23)$$

untuk pola p, dimana $p = 1, 2, \ldots, P$.

---

[13]Beale, R. dan T. Jackson, *Neural Computing : An Introduction*, IOP Publishing Ltd., Bristol, 1990, hal. 67.



Gambar 2-15
Jaringan dengan perseptron kontinyu[14]

Pengaturan bobot untuk lapisan output dinyatakan sebagai berikut :

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{2-24}$$

Yang dengan kaidah rantai, dapat ditulis sebagai :

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial (net_k)} \cdot \frac{\partial (net_k)}{\partial w_{kj}} \tag{2-25}$$

dimana

$$\frac{\partial (net_k)}{\partial w_{kj}} = y_j \tag{2-26}$$

Dan karena didefinisikan bahwa :

$$\delta_{o_k} \triangleq -\frac{\partial E}{\partial (net_k)} \tag{2-27}$$

---

[14]Zurada, Jacek M., op. cit., hal. 175.

maka diperoleh :

$$\frac{\partial E}{\partial w_{kj}} = -\delta_{o_k} y_j \qquad (2\text{-}28)$$

E adalah fungsi komposit dari net_k, sehingga persamaan (2-27) dapat ditulis lagi sebagai :

$$\delta_{o_k} = -\frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial (net_k)} \qquad (2\text{-}29)$$

dimana

$$\frac{\partial o_k}{\partial (net_k)} = f_k'(net_k) \qquad (2\text{-}30)$$

dan

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \qquad (2\text{-}31)$$

Sehingga persamaan (2-29) dapat ditulis lagi :

$$\delta_{o_k} = (d_k - o_k) f_k'(net_k), \quad untuk\ k = 1,2,\ldots,K \qquad (2\text{-}32)$$

Akhirnya diperoleh pengaturan bobot untuk jaringan lapisan tunggal :

$$\Delta w_{kj} = \eta (d_k - o_k)\ f_k'(net_k)\ y_j \qquad (2\text{-}33)$$

Pengaturaan bobot menjadi :

$$w_{kj} = w_{kj} + \Delta w_{kj} \qquad untuk\ \ k = 1,2,\ldots,K\ dan\ j = 1,2,\ldots,J \qquad (2\text{-}34)$$

Persamaan (2-33) berlaku untuk sel dengan fungsi aktivasi *f(net)* yang non-linier dan dapat diturunkan. Berikut ini tinjauan kaidah belajar delta untuk dua fungsi

aktivasi *f(net)* umum.

Untuk fungsi aktivasi kontinyu unipolar seperti pada persamaan (2-4a), diperoleh *f'(net)* :

$$f'(net) = \frac{e^{-net}}{[1+e^{-net}]^2} \qquad (2\text{-}35a)$$

Ini dapat ditulis lagi sebagai :

$$f'(net) = \frac{1}{1+e^{-net}} \cdot \frac{1+e^{-net}-1}{1+e^{-net}} \qquad (2\text{-}35b)$$

$$f'(net) = o(1-o) \qquad (2\text{-}35c)$$

Sehingga diperoleh :

$$\delta_{ok} = (d_k - o_k) o_k (1 - o_k) \qquad (2\text{-}36)$$

Untuk fungsi aktivasi kontinyu bipolar seperti pada persamaan (2-3a) dapat dinyatakan :

$$f'(net) = \frac{1}{2}(1-o)^2 \qquad (2\text{-}37a)$$

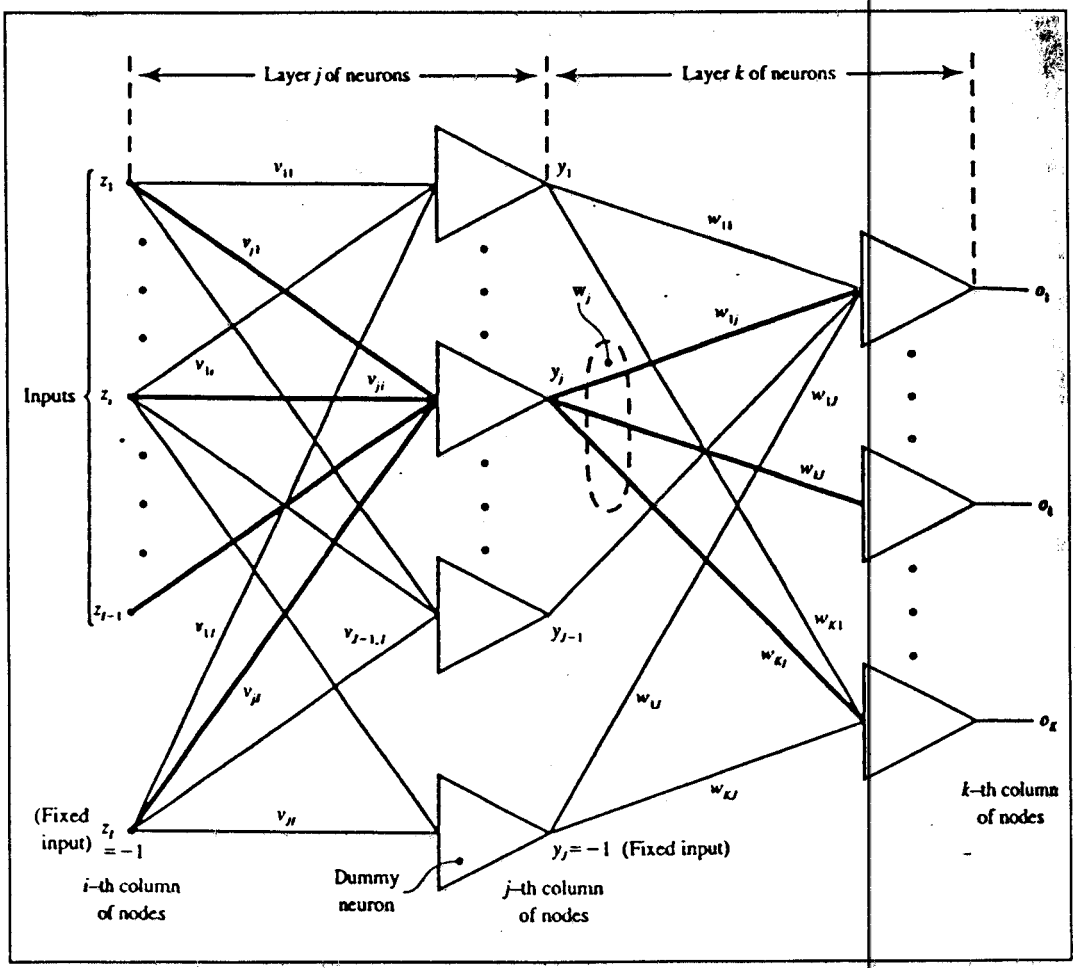
Sehingga diperoleh :

$$\delta_{ok} = \frac{1}{2}(d_{ok} - o_k)(1 - o_k^2) \qquad (2\text{-}37b)$$

2.7.3 Kaidah Belajar Delta Tergeneralisasi

Kaidah belajar delta tergeneralisasi merupakan penurunan dari kaidah belajar delta yang diterapkan pada jaringan umpan-maju lapisan jamak. Gambar

2-16 memperlihatkan jaringan yang mempunyai satu lapisan tengah.



Gambar 2-16
Jaringan dengan satu lapisan tengah[15]

Penurunan gradien negatif untuk lapisan tengah dinyatakan dengan :

$$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}}, \qquad untuk\ j = 1,2,\ldots,J\ dan\ i = 1,2,\ldots,I \qquad (2\text{-}38)$$

dan

Input untuk lapisan tengah ini adalah z_i, untuk $i = 1, 2, \ldots, I$. Maka faktor kedua

[15]ibid., hal. 182.

$$\frac{\partial E}{\partial v_{ji}} = \frac{\partial E}{\partial (net_j)} \cdot \frac{\partial (net_j)}{\partial v_{ji}} \qquad (2\text{-}39)$$

dari persamaan (2-39) :

$$\frac{\partial (net_i)}{\partial v_{ij}} = z_i \qquad (2\text{-}40)$$

dan pengaturan bobot untuk lapisan tengah dapat dinyatakan dengan :

$$\Delta v_{ji} = \eta \delta_{y_j} z_i \qquad (2\text{-}41)$$

dimana δ_{yj} adalah sinyal error dari lapisan tengah yang mempunyai output y. Sinyal error ini sama dengan :

$$\delta_{y_j} \triangleq -\frac{\partial E}{\partial (net_j)}, \quad untuk\ j = 1,2,\ldots,J \qquad (2\text{-}42)$$

Berbeda dengan eksitasi net_k pada lapisan output, yang hanya mempengaruhi output dari sel ke-k, maka net_j mempengaruhi setiap komponen error dari persamaan (2-23). Sinyal error δ_{yj} pada sel ke-j dapat dihitung sebagai berikut :

$$\delta_{y_j} = -\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial (net_j)} \qquad (2\text{-}43a)$$

dimana

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{K}\{d_k - f[net_k(y)]\}^2\right] \qquad (2\text{-}43b)$$

dan faktor kedua dari persamaan (2-43a) :

$$\frac{\partial y_j}{\partial (net_j)} = f_j'(net_j) \qquad (2\text{-}43c)$$

Perhitungan persamaan (2-43b) menghasilkan :

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K} (d_k - o_k) \frac{\partial}{\partial y_j} \{f[net_k(y)]\} \qquad (2\text{-}44a)$$

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K} (d_k - o_k) f'(net_k) \frac{\partial (net_k)}{\partial y_j} \qquad (2\text{-}44b)$$

Persamaan ini dapat disederhanakan dengan menggunakan persamaan (2-32) menjadi :

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K} \delta_{o_k} w_{kj} \qquad (2\text{-}44c)$$

Dengan menggabungkan persamaan (2-43c) dan (2-44c) diperoleh :

$$\delta_{y_j} = f_j'(net_j) \sum_{k=1}^{K} \delta_{o_k} w_{kj} \qquad (2\text{-}45)$$

Sehingga pengaturan bobot-bobot pada lapisan tengah menjadi :

$$\Delta v_{ji} = \eta f_j'(net_j) z_i \sum_{k=1}^{K} \delta_{o_k} w_{kj} \qquad (2\text{-}46)$$

2.7.4 Algoritma Belajar Error Back-Propagation

Dari kaidah perhitungan pengaturan bobot untuk lapisan output dan lapisan tengah dapat disusun suatu algoritma belajar untuk jaringan lapisan jamak. Sebagai obyek dipakai jaringan yang mempunyai satu lapisan tengah. Diberikan

pola-pola latihan sebanyak P sebagai berikut :

$$\{\mathbf{z}_1, \mathbf{d}_1, \mathbf{z}_2, \mathbf{d}_2 \ldots, \mathbf{z}_P, \mathbf{d}_P\}$$

dimana $\mathbf{z}_i$ adalah (I x 1), $\mathbf{d}_k$ adalah (K x 1), dan $i = 1, 2, \ldots, P$. Komponen ke-I dari $\mathbf{z}_i$ bernilai 1 untuk penambahan vektor input. Lapisan tengah mempunyai output **y** dengan jumlah sel J. Komponen ke-J dari **y** bernilai 1 untuk penambahan output dari lapisan tengah. Sedangkan output **o** mempunyai dimensi (K x 1). Langkah-langkah dari algoritma ini adalah sebagai berikut :

Langkah 1 : Dipilih $\eta > 0$ dan E_{maks}

Bobot **W** dan **V** diinisialisasi pada harga acak yang kecil; **W** adalah (K x J) dan **V** adalah (J x I).

$$q \leftarrow 1, \; p \leftarrow 1$$

Langkah 2 : Langkah belajar dimulai di sini.

Input diberikan dan output masing-masing lapisan dihitung :

$$z \leftarrow z_p, \quad d \leftarrow d_p$$

$$y_{pj} \leftarrow f(v_j{}^t z), \quad \textit{untuk } j = 1,2,\ldots,J$$

dimana $\mathbf{v}_j$, sebuah vektor kolom, adalah baris ke-j dari **V**, dan

$$o_{pk} \leftarrow f(w_k{}^t y), \quad \textit{untuk } k = 1,2,\ldots,K$$

dimana $\mathbf{w}_k$, sebuah vektor kolom, adalah baris ke-k dari **W**.

Langkah 3 : Menghitung vektor sinyal error δ_o dan δ_y untuk kedua lapisan.

Sinyal error untuk lapisan output :

$$\delta_{o_{pk}} = (d_{pk} - o_{pk})(1 - o_{pk})o_{pk}, \quad untuk\ k = 1,2,\ldots,K$$

Sinyal error untuk lapisan tengah :

$$\delta_{y_{pj}} = y_{pj}(1 - y_{pj})\sum_{k=1}^{K} \delta_{o_{pk}} w_{kj}, \quad untuk\ k = 1,2,\ldots,K$$

Langkah 4 : Jika $p < P$ maka $p \leftarrow p + 1$, dan kembali ke langkah 2; jika tidak, melanjutkan ke langkah 5.

Langkah 5 : Mengatur bobot pada lapisan output :

$$w_{kj} \leftarrow w_{kj} + \eta \sum_{p=1}^{P} \delta_{o_{pk}} y_{pj}, \quad untuk\ k = 1,2,\ldots,K\ dan\ j = 1,2,\ldots,J$$

Langkah 6 : Mengatur bobot pada lapisan tengah :

$$v_{ji} \leftarrow v_{ji} + \eta \sum_{p=1}^{P} \delta_{y_{pj}} z_{pi}, \quad untuk\ j = 1,2,\ldots,J\ dan\ i = 1,2,\ldots,I$$

Langkah 7 : Menghitung nilai error :

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2, \quad untuk\ k = 1,2,\ldots,K$$

Langkah 8 : Satu siklus belajar selesai.

Untuk $E < E_{maks}$ proses pelatihan selesai. Menghasilkan **W**, **V**, q, dan E.

Jika $E > E_{maks}$, maka $E \leftarrow 0$, $p \leftarrow 1$, $q \leftarrow q + 1$ dan menuju ke langkah 2 untuk memulai siklus belajar yang baru.

2.7.5 Error

Pada pembahasan di atas, error dihitung secara kumulatif, artinya error dihitung setelah semua pola latihan dimasukkan dan dihitung aktivasinya masing-masing. Ini dinyatakan dengan :

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2 \qquad (2\text{-}47)$$

Error ini penjumlahan dari P error yang dihitung untuk masing-masing pola. Definisi error semacam ini tidak dapat digunakan untuk membandingkan jaringan-jaringan dengan jumlah pola latihan yang berbeda dan sel output yang berbeda. Jaringan dengan jumlah sel output K yang dilatih dengan pola latihan yang banyak akan menghasilkan error kumulatif yang besar. Demikian pula dengan jaringan yang mempunyai sel output yang banyak yang dilatih dengan pola latihan yang sama. Maka penghitungan error yang lebih sesuai dapat dinyatakan sebagai berikut :

$$E_{rms} = \frac{1}{PK}\sqrt{\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})} \qquad (2\text{-}48)$$

2.7.3 Faktor-faktor dalam Pelatihan EBP

2.7.3.1 Bobot Awal

Bobot-bobot dalam jaringan yang dilatih biasanya diinisialisasi pada nilai acak yang kecil. Inisialisasi ini sangat mempengaruhi hasil akhir. Jika semua bobot awal diberi harrga yang sama, jaringan tidak akan terlatih dengan benar. Jaringan mungkin saja akan gagal untuk belajar serangkaian contoh-contoh pelatihan dengan errornya yang tetap atau bahkan error akan naik begitu proses pelatihan diteruskan. Dari penelitian-penelitian yang telah dilakukan diperoleh bahwa melanjutkan pelatihan di luar daerah error tertentu akan menghasilkan bobot yang tidask diinginkan. Ini mengakibatkan error naik dan kualitas pemetaan jaringan akan turun. Untuk mengatasi masalah ini, jaringan harus dilatih dengan bobot acak yang lain.

2.7.3.2 Konstanta Belajar (Learning Constant)

Efektifitas dan konvergensi dari algoritma error-back propagation sangat tergantung pada nilai konstanta belajar (η). Secara umum, nilai optimal dari η tergantung pada masalah yang sedang diselesaikan. Seperti yang telah diketahui bahwa back-error propagation didasarkan pada penurunan gradien error. Penurunan gradien ini merupakan metode yang efisien untuk menghasilkan nilai bobot yang meminimkan error. Tetapi seringkali bentuk permukaan error membuat prosedur ini menjadi lambat untuk kovergen.

Pada minima yang lebar atau nilai gradiennya kecil, nilai η yang besar akan menghasilkan konvergensi yang lebih cepat. Tetapi pada minima yang curam

dan sempit, harus dipilih nilai η yang kecil untuk menghindari overshoot. Dari kenyataan tersebut dapat disimpulkan bahwa η harus dipilih berdasarkan percobaan.

Hanya dengan nilai η yang kecil yang akan memberikan penurunan gradien dengan benar. Namun ini harus dibayar dengan jumlah iterasi yang bertambah banyak untuk mendapatkan hasil yang memuaskan.

Meskipun pemilihan konstanta belajar ini sangat tergantung pada permasalahan dan arsitektur jaringan, tetapi ada rentang nilai tertentu yang diperoleh dari banyak eksperimen sukses yang telah dilakukan selama ini, yakni antara 10^{-3} sampai dengan 10.

2.7.3.3 Metode Momentum

Tujuan dari metode momentum adalah untuk mempercepat konvergensi dari algoritma Error Back-Propagation. Prinsip dari metode ini adalah menambahkan sebagian dari perubahan bobot yang sebelumnya pada pengaturan bobot yang sedang berlangsung. Ini dapat dirumuskan dengan :

$$\Delta w(t) = -\eta \nabla E(t) + \alpha \Delta w(t-1)$$

dimana t dan t-1 menunjukkan langkah pelatihan yang sedang berlangsung dan langkah sebelumnya. α adalah konstanta momentum yang berupa bilangan positif. Suku kedua pada ruas kanan menunjukkan sebagian dari perubahan bobot yang sebelumnya, disebut suku momentum. Untuk N langkah iterasi dengan menggunakan metode momentum, perubahan bobot yang sekarang dapat

dinyatakan sebagai berikut :

$$\Delta w(t) = -\eta \sum_{n=0}^{N} \alpha^n \nabla E(t-n)$$

biasanya α dipilih antara 0,01 dan 0,8.

2.8 Aspek-aspek Jaringan Saraf Tiruan

2.8.1 Konvergensi

Jika pelatihannya hasil dengan baik, jaringan akan memberikan jawaban yang benar. Ini memerlukan suatu ukuran kuantitatif dari pelatihan. Ukuran yang biasa digunakan adalah error RMS (Root Mean Square). Ukuran ini menunjukkan sampai sejauh mana jaringan akan memberikan jawaban yang benar.

Jawaban yang diberikan jaringan bukanlah ya atau tidak. Karena nilai target dari jaringan berupa bilangan real, demikian pula outputnya. Untuk itu diperlukan adanya nilai batas untuk menentukan apakah jawaban yang diberikan output jaringan bernilai benar.

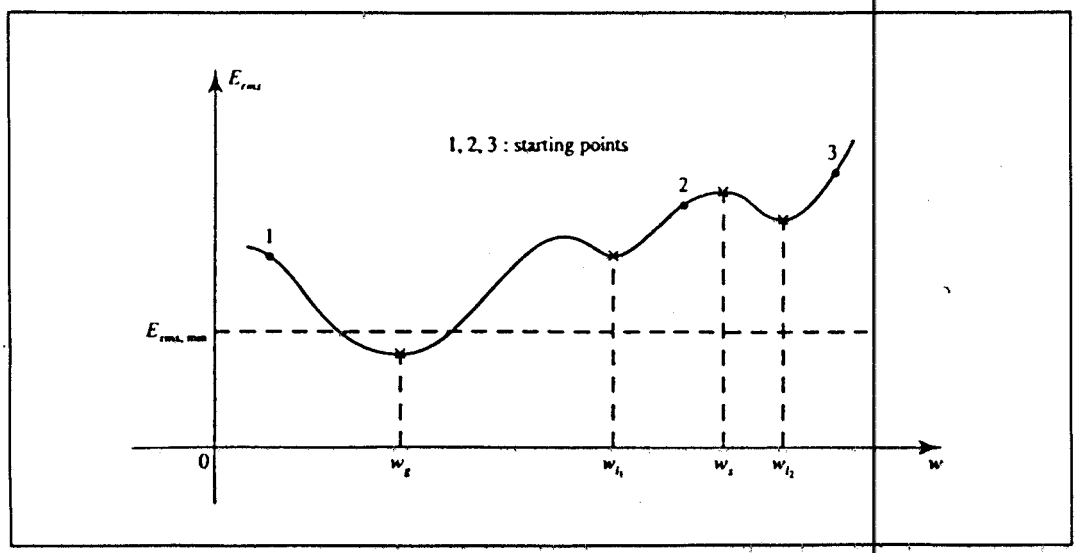
Konvergensi adalah proses dimana nilai RMS semakin mendekati nol. Konvergensi tidak selalu mudah dicapai karena kadang-kadang jaringan terjebak di minimum lokal dan pelatihan terpaksa dihentikan. Jaringan Back-Propagation biasanya mudah untuk konvergen ke nilai RMS yang bagus jika pola-pola input latihannya benar-benar jelas perbedaannya.

2.8.2 Minimum Lokal

Salah satu masalah dalam minimisasi nilai error RMS adalah adanya

minimum lokal dalam fungsi error. Gambar 2-17 memperlihatkan potongan dari fungsi error dalam dimensi bobot tunggal. Terlihat bahwa error adalah fungsi non-negatif dari variabel bobot. Pemetaan yan ideal akan mereduksi E_{rms} ke arah 0.

Fungsi error pada Gambar 2-17 mempunyai satu minimum global (pada w_g) di bawah nilai E_{rms} minimum, dan juga mempunyai dua minimum lokal pada w_{l1} dan w_{l2}, dan satu titik stasioner pada w_s. Prosedur belajar akan berhenti secara dini jika dimulai dari titik 2 atau 3; sehingga jaringan yang dilatih tidak memberikan performa sesuai dengan yang diinginkan.



Gambar 2-17
E_{rms} sebagai fungsi bobot tunggal[16]

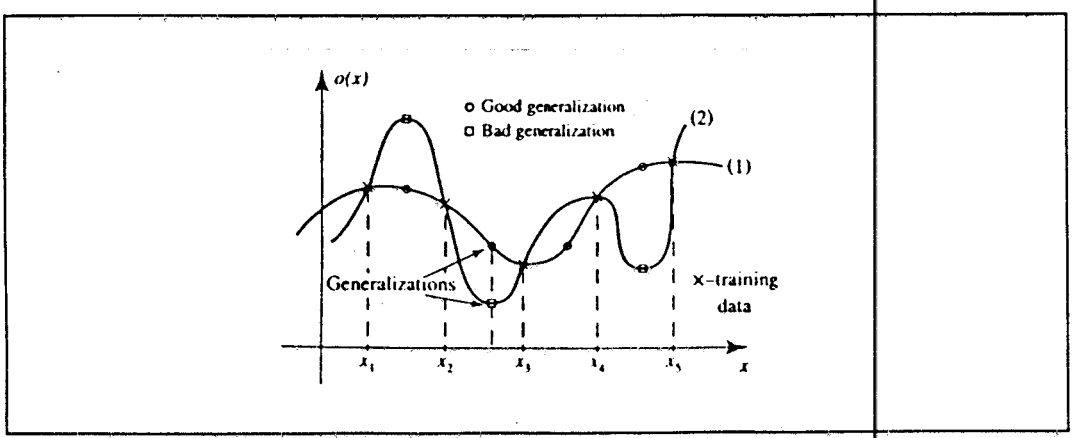
2.8.2 Generalisasi

Satu ciri khas dari jaringan saraf tiruan adalah kemampuannya dalam generalisasi, yakni dapat mengklasifikasikan pola yang tidak pernah diberikan sebelumnya. Multilayer perceptron melakukan generalisasi dengan jalan

[16]ibid., hal. 207.

mendeteksi ciri-ciri khas dari pola input yang telah diberikan dan kemudian meng-kode-kan dalam bentuk bobot. Dengan demikian pola yang tidak diketahui diklasifikasikan dengan pola lainnya yang mempunyai kesamaan ciri. Ini berarti bahwa input yang mengandung noise dapat diklasifikasikan dengan melihat kesamaannya dengan pola input yang asli.

Secara umum, jaringan saraf tiruan bagus pada interpolasi, tetapi tidak begitu bagus pada ekstrapolasi. Mereka mampu mendeteksi pola-pola yang ada pada input-input, dan memperbolehkan kedudukan menengah yang belum pernah dilihat sebelumnya. Tetapi, input-input yang merupakan perluasan dari rentang pola-pola yang dilatihkan akan lebih sulit diklasifikasikan, karena hanya ada sedikit bagian yang dapat dibandingkan.



Gambar 2-18
Generalisasi dari suatu jaringan[17]

Gambar 2-18 mengilustrasikan generalisasi yang bagus dan yang buruk pada data-data pola yang baru dan berada di antara pola-pola latihan. Jaringan dilatih menggunakan data-data x_1 sampai x_5.

[17]ibid., hal. 54.

BAB III

TRANSPUTER

3.1 Pendahuluan

Kemampuan komputer yang ada saat ini sangat mengagumkan dengan kecepatannya yang semakin tinggi. Tetapi untuk aplikasi-aplikasi tertentu yang ditujukan untuk menghasilkan model dunia nyata yang lebih baik komputer yang ada masih belum cukup memadai. Sebagai contoh di sini adalah membuat komputer untuk melakukan sesuatu selayaknya otak manusia, seperti pengenalan suara, analisa citra dan pengenalan pola, pengelihatan buatan, dan pemahaman bahasa. Untuk itu dibutuhkan komputer dengan kemampuan yang lebih tinggi lagi.

Para pakar berusaha untuk mengembangkan kecepatan komputer tetapi mereka menemui beberapa keterbatasan. Yang paling utama adalah batasan fisik yakni kecepatan cahaya. Untuk mendekati kecepatan ini saja adalah tidak mungkin. Salah satu upaya mengatasi masalah kecepatan yaitu penemuan sirkuit *photonic* yang menggunakan cahaya kilat, termasuk arus listrik, untuk mentransformasikan informasi digital melalui serat optik. Serat optik ini hampir tanpa desipasi panas dan tidak membutuhkan insulasi elektris. Konsep yang lebih radikal dari sirkuit *photonic* adalah *biochip*, yaitu komponen tiga dimensi yang dibuat dari molekul-molekul organik. Molekul-molekul ini dapat dipaketkan bersama-sama dalam densitas yang lebih besar dari komponen semikonduktor.

Pengembangan ke arah tersebut sangat menarik tetapi tidak dapat

diwujudkan untuk pemakaian komersial saat ini juga memerlukan biaya yang sangat tinggi untuk tiap unit dan sistem. Super komputer menggunakan Gallium Arsenide atau supercooled sirkuit CMOS yang membutuhkan memori yang sangat cepat dan trilyunan rupiah untuk memiliki serta mengoperasikan.

Berdasarkan kendala-kendala diatas, multiproses adalah salah satu alternatif yang paling mungkin. *Transputer* adalah prosesor yang didesain untuk dapat bekerja secara paralel dengan satu atau lebih transputer lainnya. Transputer dapat meningkatkan kecepatannya dengan mengeksekusi bagian-bagian dari sebuah persoalan secara bersama-sama. Sebuah transputer menangani bagian yang berbeda dari sebuah persoalan dengan cara mengeksekusi tiap bagian secara bergantian dengan cepat atau sebuah jaringan dari transputer menangani bagian-bagian tersebut secara simultan dengan mendistribusikannya ke tiap-tiap transputer. Untuk membagi suatu masalah menjadi bagian-bagian yang akan diproses oleh masing-masing transputer secara efisien dan efektif diperlukan pembagian beban kerja yang seimbang pada tiap transputer (*load balancing*).

Pada bab ini akan dibahas mengenai arsitektur umum dari transputer baik konfigurasi dasar, yaitu fasilitas-fasilitas standar pada setiap transputer, tipe-tipe yang ada pada keluarga transputer, sistem pengalamatan maupun register-register pada prosesornya. Kemudian dilanjutkan dengan pembahasan mengenai proses sekuen dan konkuren. Berikutnya adalah mengenai konsep yang paling penting dalam dunia transputer, yaitu *channel* yang terdiri atas channel internal atau *soft channel* dan channel eksternal atau *hard channel* yang lebih dikenal sebagai *link*. Pembahasan mengenai link dilanjutkan lebih detil pada bagian komunikasi antar

proses. Selanjutnya ada pembahasan khusus pada transputer tipe T805 yang dipakai dalam tugas akhir ini. Dan akhir dari bab menampilkan piranti perantara komunikasi transputer dengan dunia luar baik periferal (I/O) maupun prosesor jenis lain. Piranti ini disebut dengan Link Adaptor.

3.2 Arsitektur Transputer

3.2.1 Konfigurasi Dasar

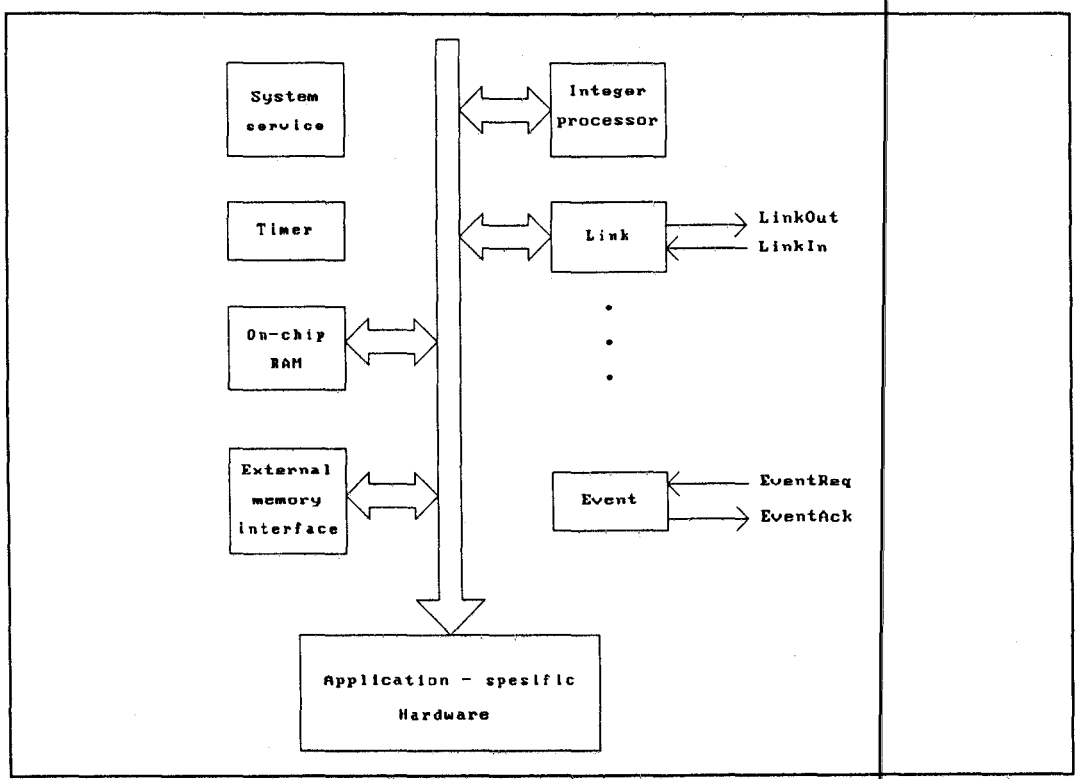
Komponen dasar dari semua mikroprosesor adalah eksekusi instruksi dan unit dekoding, yang sering disebut dengan Central Processing Unit (CPU) dan External Memory Interface (EMI). CPU membaca instruksi dari memori melalui EMI dan melaksanakannya secara sekuensial (berurutan). Desain chip modern ditujukan untuk menjadikan kombinasi prosesor standar dan interface memori ini berjalan secepat mungkin. Hal ini dilakukan dengan jalan melengkapi chip dengan fungsi-fungsi yang sebelumnya membutuhkan ko-prosesor eksternal. Fungsi-fungsi ini adalah manajemen memori, aritmatika floating-point, serta cache instruksi dan data.

Perangkat keras manajemen memori memetakan ruang alamat virtual ke memori fisik. Tiap proses dapat mempunyai ruang alamat terpisah dan pemetaan yang berbeda ke memori fisik, sehingga memori yang digunakan oleh satu proses dapat dicegah dari peng-aksesan oleh proses lain. Perangkat keras dekoding alamat dapat dibuat berjalan jauh lebih cepat jika modul manajemen memori diintegrasikan ke dalam chip yang sama dengan CPU.

Tambahan yang umum pada chip adalah cache, yakni peletakan data atau

kode yang sering diakses ke dalam chip. Keluarga transputer menggunaakn pendekatan lain, yakni dengan adanya on-chip RAM (memori internal) yang dapat dipakai sama seperti memory eksternal tetapi dengan kecepatan 2 atau 3 kali lebih cepat. Tidak seperti cache, pada transputer dibutuhkan pemrograman yang menentukan apakah RAM ini berisi kode, data atau keduanya.

Fungsi utama yang merupakan kelebihan transputer adalah dengan adanya komunikasi on-chip. Dengan fasilitas ini transputer mempunyai kemampuan untuk komunikasi secara langsung dengan transputer lain.



Gambar 3-1
Konfigurasi dasar transputer[18]

[18]Graham, Ian dan Tim King, *The Transputer Handbook*, Prentice Hall International (UK), 1990, hal. 8.

Fasilitas-fasilitas standar yang ada pada transputer dapat disimpulkan sebagai berikut :

- Prosesor integer berkecepatan tinggi dengan penjadwal proses terkode-mikro
- Memori statis cepat dalam chip
- 2 atau 4 link komunikasi
- Timer internal
- Interface memori eksternal

3.2.2 Keluarga Transputer

Keluarga transputer terdiri dari tiga kelompok utama : seri T2 (16-bit), seri T4 (integer 32-bit) dan seri T8 (32-bit dengan *Floating Point Unit (FPU)* 32/64 bit). Grup ini hanya membedakan prosesor dalam besar RAM internal, banyaknya link, detil instruction set dan tipe interface memori. Walaupun demikian semua transputer memiliki model arsitektur dasar yang serupa. Tabel 3-1 memperlihatkan anggota dari keluarga transputer.

Tabel 3-1
Keluarga Transputer (Desember 1989)[19]

	transputer 16 bit			
	T212	T222	T225	M212
Panjang word	16	16	16	16
Internal RAM	2K	4K	4K	2K
Jumlah Link	4	4	4	2
Instruksi lanjut	n	y	y	n
Instruksi Debug	n	n	y	n

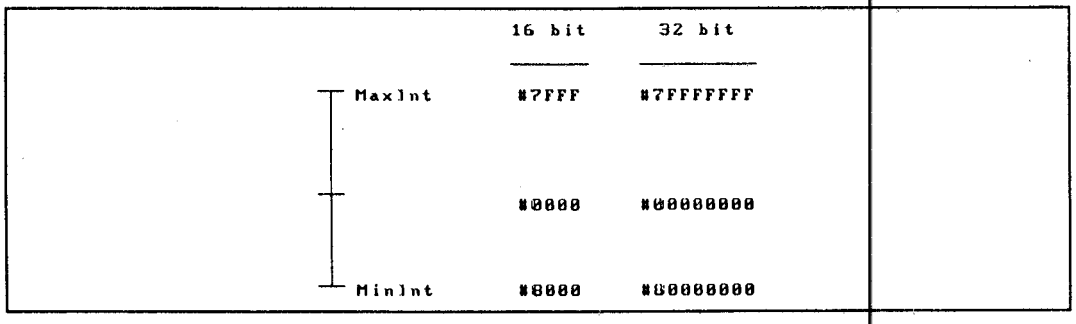
[19]ibid., hal. 16.

	transputer 32 bit					
	T400	T414	T425	T800	T801	T805
Panjang word	32	32	32	32	32	32
Internal RAM	2K	2K	4K	4K	4K	4K
Jumlah Link	2	4	4	4	4	4
Hardware FPU	n	n	n	y	y	y

3.2.3 Pengalamatan memori

3.2.3.1 Rentang alamat

Semua transputer dialamati menurut byte. Gabungan dari byte membentuk word. Pada tipe T2xx satu word terdiri atas 2 byte, dan pada tipe T8xx dan T4xx satu word adalah 4 byte. Panjang alamat selalu ditulis dalam word (integer). Transputer 16 bit (T2xx) mempunyai ruang alamat sebesar 64 Kbyte. Transputer 32 bit (T8xx dan T4xx) mempunyai ruang alamat 4 Gbyte.

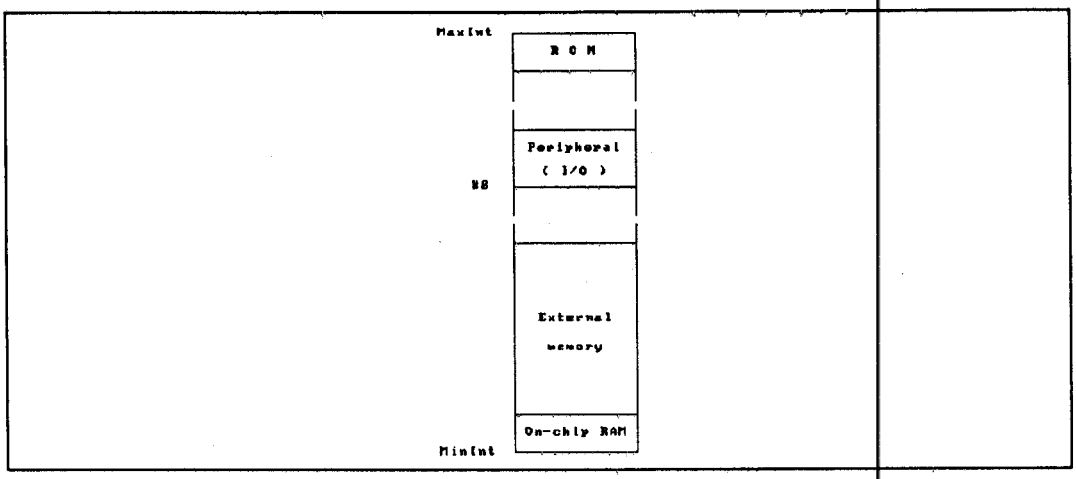


Gambar 3-2
Rentang alamat transputer

Pengalamatan memori transputer merupakan pengalamatan linier. Rentang alamatnya tidak seperti pada prosesor umumnya, yang dimulai dari #00..00 sampai #FF..FF. Tetapi alamat transputer merupakan bilangan integer bertanda yang dimulai dari #80..00 samapi dengan #7F..FF, dengan alamat #00..00 berada di tengah-tengah. #80..00 disebut dengan **MinInt** dan #7F..FF **MaxInt**.

3.2.3.2 Peta memori

Gambar 3-3 memperlihatkan peta memori yang diterapkan pada transputer.



Gambar 3-3
Peta memori transputer

Memori internal (2 atau 4 Kbyte) berada di bagian paling bawah dari rentang alamat, dimulai pada **MinInt**. Peletakan ini sudah ditetapkan oleh perangkat keras transputer. Memori intrnal ini sangat cepat dan diakses dalam satu siklus prosesor. Pemakaian memori internal yang baik akan meningkatkan performa. Memori internal ini biasanya digunakan sebagai stack. Memori internal dapat di-non-aktif-kan dengan pin **DisableIntRam**.

Memori eksternal terletak tepat di atas memori internal. Akses ke memori eksternal lebih lambat daripada ke memori internal. Sistem akan berjalan lebih cepat bila pemilihan letak data maupun kode dilakukan dengan tepat.

Jika diinginkan sistem yang berdiri sendiri (*stand-alone*), maka dibutuhkan adanya ROM. ROM ini harus diletakkan di bagian paling atas dari rentang memori. Hal ini karena alamat reset (alamat dari instruksi yang pertama kali akan

dijalankan pada saat sistem di-reset) dari prosesor adalah #7F..FF.

Sedangkan piranti input output (I/O) atau periferal diletakkan di tengah-tengah, yaitu mulai dari alamat #00..00.

3.2.3.3 Memori yang dicadangkan

Pada bagian bawah dari rentang alamat, yang dimulai dari **MinInt** sampai dengan **MemStart**, terdapat blok memori yang dicadangkan. Memori ini digunakan oleh prosesor untuk tujuan-tujuan khusus. Pemakai dapat membaca isi memori ini, tetapi menulis ke alamat memori ini akan sangat berbahaya. Ukuran memori ini tergantung pada tipe transputer, 36 byte pada T225 dan 112 byte pada T805.

Memori ini terletak pada memori internal (on-chip RAM), untuk memastikan tetap adanya memori fisik walaupun tidak ada memori eksternal. Artinya transputer tetap dapat menjalankan program dengan tanpa memori eksternal. Ruang memori yang dicadangkan ini digunakan untuk tiga tujuan : 9 word pertama digunakan oleh link dan Event, 2 word berikutnya digunakan oleh timer dan sisanya adalah daerah penyimpanan interupsi. Tabel 3-2 berisi daftar memori yang dicadangkan.

Tabel 3-2
Lokasi memori yang dicadangkan (T225 dan T805)[20]

Alamat Word	Nama	Kegunaan
MintInt+28	MemStart	T805,T800,T425,T225
...		dicadangkan untuk
...		instruksi lanjutan
MinInt+18	MemStart	T414,T212,M212
MinInt+17	EregIntSaveLoc	
MinInt+16	STATUSIntSaveLoc	
MinInt+15	CregIntSaveLoc	
MinInt+14	BregIntSaveloc	daerah penyimpan
MinInt+13	AregIntSaveloc	register
MinInt+12	IptrIntSaveloc	
MinInt+11	WdescIntSaveLoc	
MinInt+10	TptrLoc1	timer prioritas rendah
MinInt+9	TPtrLoc0	timer prioritas tinggi
MinInt+8	Event	kanal event
MinInt+7	Link3Input	
MinInt+6	Link2Input	
MinInt+5	Link1Input	
MinInt+4	Link0Input	kontrol word
MinInt+3	Link3Output	link
MinInt+2	Link2Output	
MinInt+1	Link1Output	
MinInt	Link0Output	

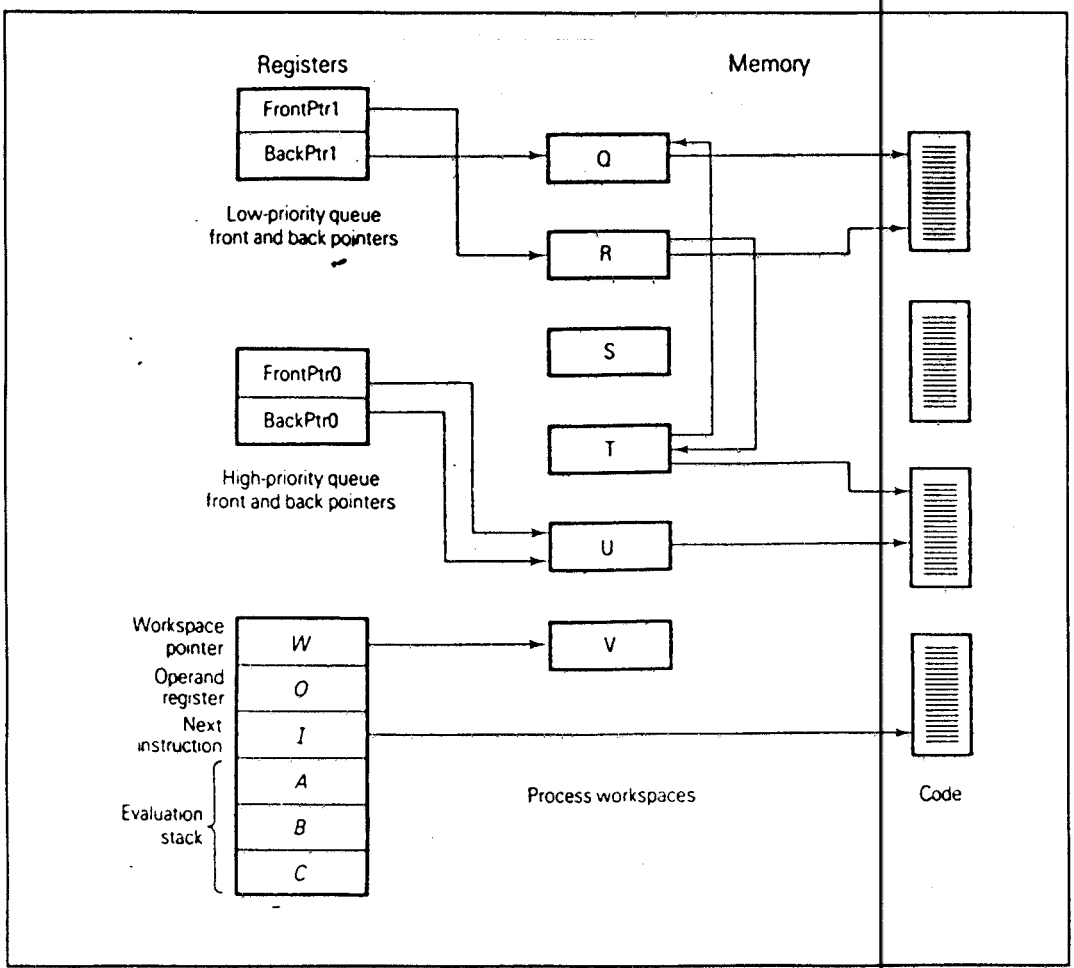
3.2.4 Register-register Transputer

Prosesor dari transputer mempunyai hanya sejumlah kecil register. Ada 6 register yang digunakan dalam pelaksanaan proses sekuen. Keenam register tersebut yakni : A, B, C, W, I dan O. Register A, B dan C merupakan register evaluasi yang berupa stack. A, B dan C adalah *source dan destination register* untuk sebagian besar operasi aritmatika dan logika. Hanya register A yang dapat diakses langsung, baik dibaca maupun ditulisi. Menulis ke register A akan mem-*push* isi register A yang lama ke register B, dan isi register B yang lama ke register C, dan isi C yang lama akan hilang. Mengambil nilai dari register A akan

[20]ibid., hal. 15.

mem-*pop* isi B ke A, dan isi C ke B.

Register W, I dan O merupakan register kontrol proses sekuen. Register W berisi Workspace pointer yang menunjuk alamat terbawah daerah penyimpanan data dimana variabel lokal dari suatu proses berada. Register I berisi Instruction pointer yang menunjuk instruksi berikutnya yang akan dilaksanakan. Sedangkan register O berisi operand dari instruksi yang sedang dilaksanakan.



Gambar 3-4
Register-register transputer[21]

[21]ibid,. hal 12.

Register-register FrontPtr (FPtr) dan BackPtr (Bptr) digunakan dalam proses-proses konkuren. Alamat dari proses urutan pertama atau terdepan terletak pada FPtr, dan sebaliknya BPtr menunjuk proses di urutan terakhir. Ada dua macam FPtr yaitu FPtr0 dan FPtr1, demikian pula ada dua BPtr, BPtr0 dan BPtr1. FPtr0 dan BPtr0 digunakan untuk daftar eksekusi proses dengan prioritas tinggi. FPtr1 dan BPtr1 untuk daftar prioritas rendah.

3.3 Penjadwal Proses (Process Scheduler)

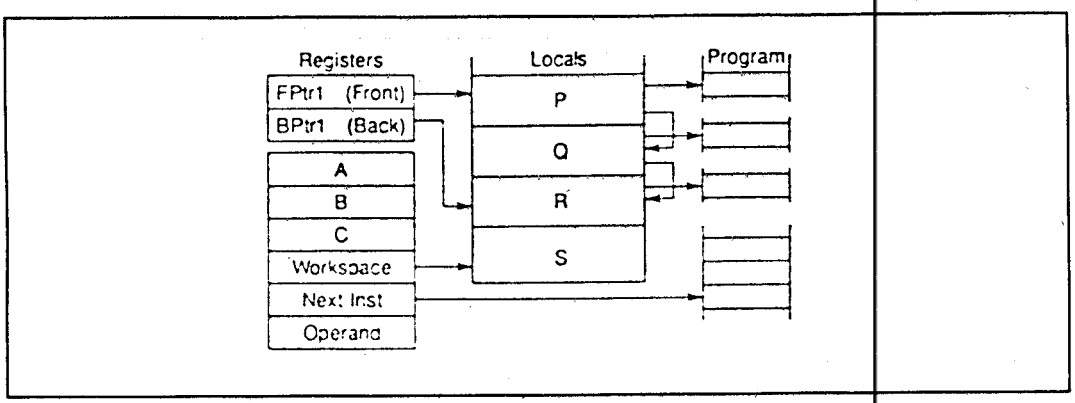
Sebuah proses dimulai untuk menjalankan sujumlah instruksi. Proses ini dapat berhenti dengan dua kemungkinan, proses belum selesai dan proses sudah selesai. Biasanya sebuah proses merupakan serangkaian instruksi. Transputer dapat menjalankan banyak proses secara paralel (konkuren). Proses-proses tersebut dapat diberi prioritas tinggi atau rendah.

Prosesor transputer mempunyai sebuah penjadwal proses terkodemikro yang memungkinkan sejumlah proses dijalankan bersama-sama. Ini menggantikan kebutuhan akan kernel perangkat lunak .

Pada suatu saat, sebuah proses konkuren dalam keadaan aktif atau non-aktif. Suatu proses aktif bila sedang dieksekusi atau sedang dalam daftar tunggu untuk dieksekusi. Sedangkan suatu proses disebut non-aktif bila sedang siap menerima input, siap mengirim data (output) atau menunggu selama selang waktu tertentu.

Penjadwal bekerja sedemikian hingga proses yang tidak aktif tidak memakan waktu prosesor. Proses-proses aktif yang sedang menunggu untuk

dieksekusi dimasukkan dalam dua daftar berantai dari daerah-kerja (*workspace*) proses, satu untuk proses-proses prioritas tinggi dan satu untuk prioritas rendah. Masing-masing daftar diimplementasikan menggunakan dua register, satu untuk proses pertama dan satu untuk proses terakhir, seperti dijelaskan pada sub-bab 3.2.4. Dalam Gambar 3-5, proses S sedang dieksekusi; P, Q dan R proses aktif yang sedang menunggu eksekusi.



Gambar 3-5
Daftar proses berantai[22]

3.4 Channel

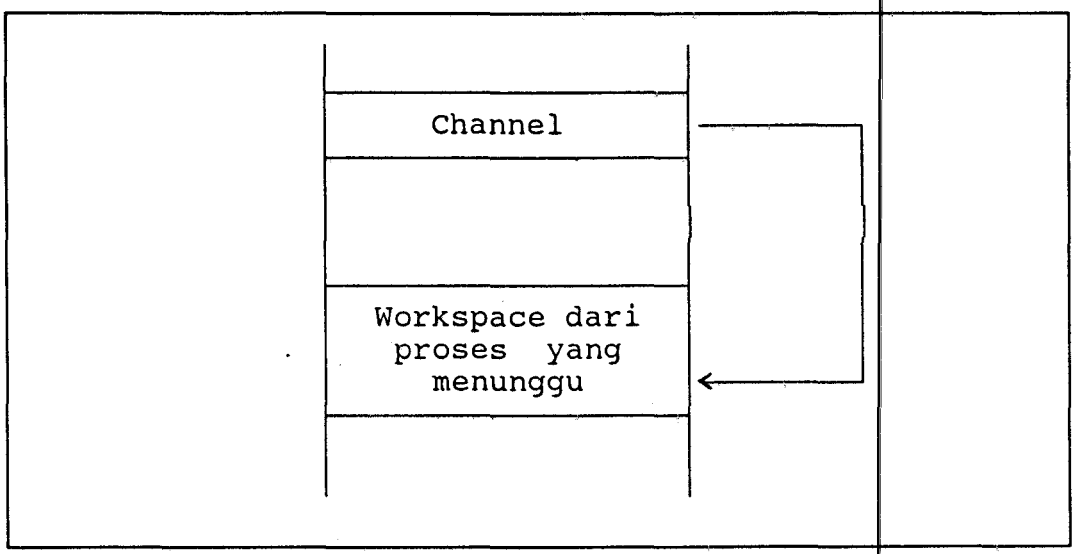
Jalur komunikasi antara dua proses dikenal dengan sebutan *Channel*. Dalam transputer, channel merupakan konsep yang penting. Channel memberikan komunikasi yang dibutuhkan dalam pemrograman real-time dan parallel. Channel bekerja dengan baik pada proses-proses dalam satu transputer maupun antar transputer. Channel antara proses-proses dalam transputer yang sama disebut *soft channel* atau channel internal. Hal ini dilakukan oleh prosesor dengan jalan meng-copy data dari daerah data satu proses ke daerah data proses yang lain. Channel

[22]----, *Transputer Technical Spesifications*, Computer System Architects, USA, 1990, hal. 10.

antara proses-proses pada transputer yang berbeda disebut *hard channel* atau channel eksternal. Ini dilakukan oleh port komunikasi transputer yang dikenal sebagai *link*.

3.4.1 Channel Internal

Satu channel hanya dapat digunakan oleh dua proses parallel. Proses yang satu mengirim dan yang lain menerima. Proses penerima yang sudah siap harus menunggu sampai proses pengirim juga siap. Jika prosesor memberi instruksi input atau output, prosesor memberitahu apakah proses di sisi yang lain (proses kedua) telah siap. Jika belum siap, proses tersebut harus menunggu. Pointer dari proses ini disimpan, dan jika proses yang lain telah siap, prosesor memberi tahu bahwa proses pertama sedang menunggu, sehingga komunikasi dapat dimulai. Jika transfer data telah selesai, kedua proses diaktifkan kembali.



Gambar 3-6
Implementasi channel

Channel diimplementasikan dalam memori sebagai word tunggal. Word ini digunakan untuk menyimpan alamat dari proses yang sedang menunggu (Gambar 3-6). Hanya satu proses yang dapat menunggu pada satu saat. Jika kedua proses telah siap kemudian dilakukan komunikasi.

Dengan melihat word channel, prosesor dapat memberitahu apakah ada proses yang sedang menunggu. Tidak seperti port konvensional, channel tidak berisi data. Jika sebuah proses mengirim lewat channel, proses ini tidak menulis ke channel. Word channel hanya digunakan sebagai pointer proses yang sedang menunggu. Pada prinsipnya, dua proses memiliki data terpisah dalam ruang terpisah di memori. Komunikasi dilakukan dengan meng-copy langsung data dari proses yang satu ke proses yang lain.

Sebelum digunakan untuk komunikasi channel haru diinisialisasi terlebih dahulu, rutin-rutin berikut digunakan untuk alokasi dan inisialisasi channel :

- *int ChanReset(Channel *c)*
- *Cannel *ChanAlloc()*

ChanReset() untuk mereset channel dan ***ChanAlloc()*** memberikan pointer dari channel.

Ada 6 rutin yang disediakan untuk komunikasi melalui channel :

- *ChanOut(Channel *c, (char *)ptr, int n)* : mengirim n byte data.
- *ChanOutChar(Channlel *c, char data)* : mengirim 1 byte data.
- *ChanOutInt(Channel *c, int data)* : mengirim 1 word data.
- *ChanIn(Channlel *c, (char *), int n)* : menerima n byte data.

- *int ChanInInt(Channel *c)* : menerima 1 word data.
- *char ChanInChar(Channlel *c)* : menerima 1 byte data.

dimana :

c = pointer channel

ptr = pointer data

3.4.2 Channel Eksternal (Link)

Channel eksternal atau link merupakan jalur hubungan transputer dengan dunia luar, baik dengan transputer lain, periferal maupun prosesor lain. Link merupakan hubungan titik ke titik *(point-to-point)* dan mempunyai dua arah. Masing-masing link merupakan dua channel dengan arah yang berlawanan. Transputer mempunyai dua atau empat buah link. Masing-masing link mempunyai dua pin, yaitu **LinkIn** dan **LinkOut**. **LinkIn** untuk sinyal-sinyal yang masuk dan **LinkOut** untuk keluar. Sinyal-sinyal ini aktif dengan logik tinggi dan mempunyai level TTL.

Link ini sangat fleksibel dan dapat digunakan untuk berbagai tujuan, untuk interface dengan periferal melalui *link adaptor* maupun komunikasi dengan prosesor lain. Tapi yang paling penting dalam hal ini link digunakan untuk berkomunikasi dengan transputer lain.

Channel eksternal tidak membutuhkan inisialisasi maupun alokasi lagi, karena sudah didefinisikan pada alamat tertentu. Untuk transputer dengan empat link, definisi kedelapan pointer channel tersebut yaitu :

*#define LINK0OUT ((Channel *) 0x80000000)*

*#define LINK0OUT ((Channel *) 0x80000004)*

*#define LINK0OUT ((Channel *) 0x80000008)*

*#define LINK0OUT ((Channel *) 0x8000000C)*

*#define LINK0OUT ((Channel *) 0x80000010)*

*#define LINK0OUT ((Channel *) 0x80000014)*

*#define LINK0OUT ((Channel *) 0x80000018)*

*#define LINK0OUT ((Channel *) 0x8000001C)*

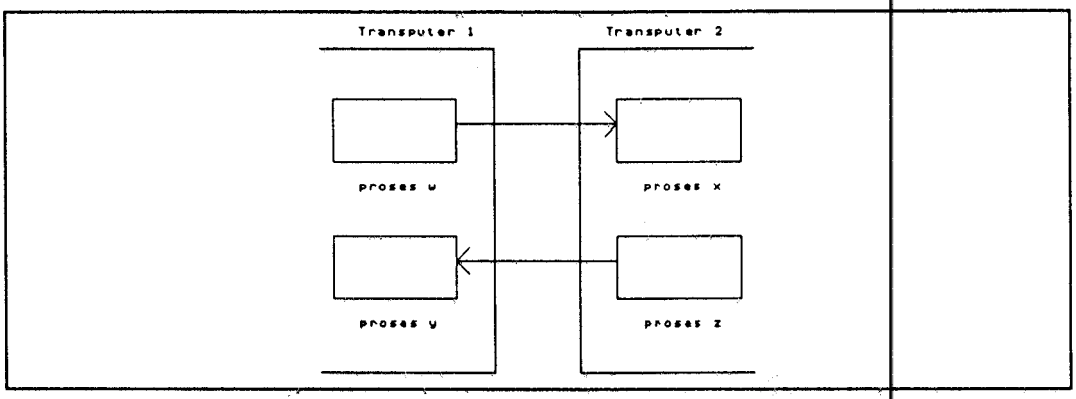
Maka instruksi untuk mengirim data sebanyak 10 byte dengan pointer awal data ptr melalui LINK0 adalah sebagai berikut :

*ChanOut(LINK0OUT, (char *)ptr, 10).*

3.5 Komunikasi antar transputer

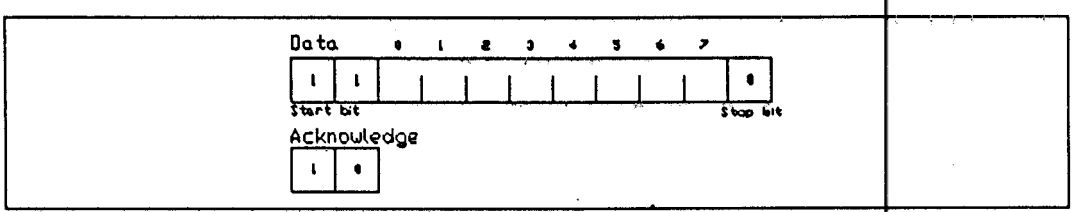
Komunikasi antara dua transputer harus sinkron. Oleh karena itu masing-masing pesan harus di-*acknowledge*. Komunikasi ini dilakukan dengan jalan menghubungkan link interface suatu transputer dengan link interface transputer yang lain dengan dua jalur sinyal, dimana data dikirim secara serial.

Masing-masing pesan dikirimkan sebagai serangkaian komunikasi byte tunggal, sehingga hanya membutuhkan adanya buffer byte tunggal pada transputer penerima untuk memastikan bahwa tidak ada informasi yang hilang.



Gambar 3-7
Komunikasi link antar proses[23]

Masing-masing byte dikirimkan dengan urutan : start bit diikuti satu bit, kemudian delapan bit data, dan diakhiri dengan stop bit. Setelah itu pengirim menunggu sampai acknowledge diterima, yang berupa start bit yang diikuti zero bit (Gambar 3-8). Acknowledge ini menandakan bahwa sebuah proses telah menerima data tersebut, dan bahwa link penerima siap menerima byte yang lain. Link pengirim mendaftar ulang (*reschedule*) pengirim hanya setelah acknowledge untuk byte terakhir dari pesan yang dikirim telah diterima.



Gambar 3-8
Link protokol[24]

3.6 Karakteristik Transputer IMS T805

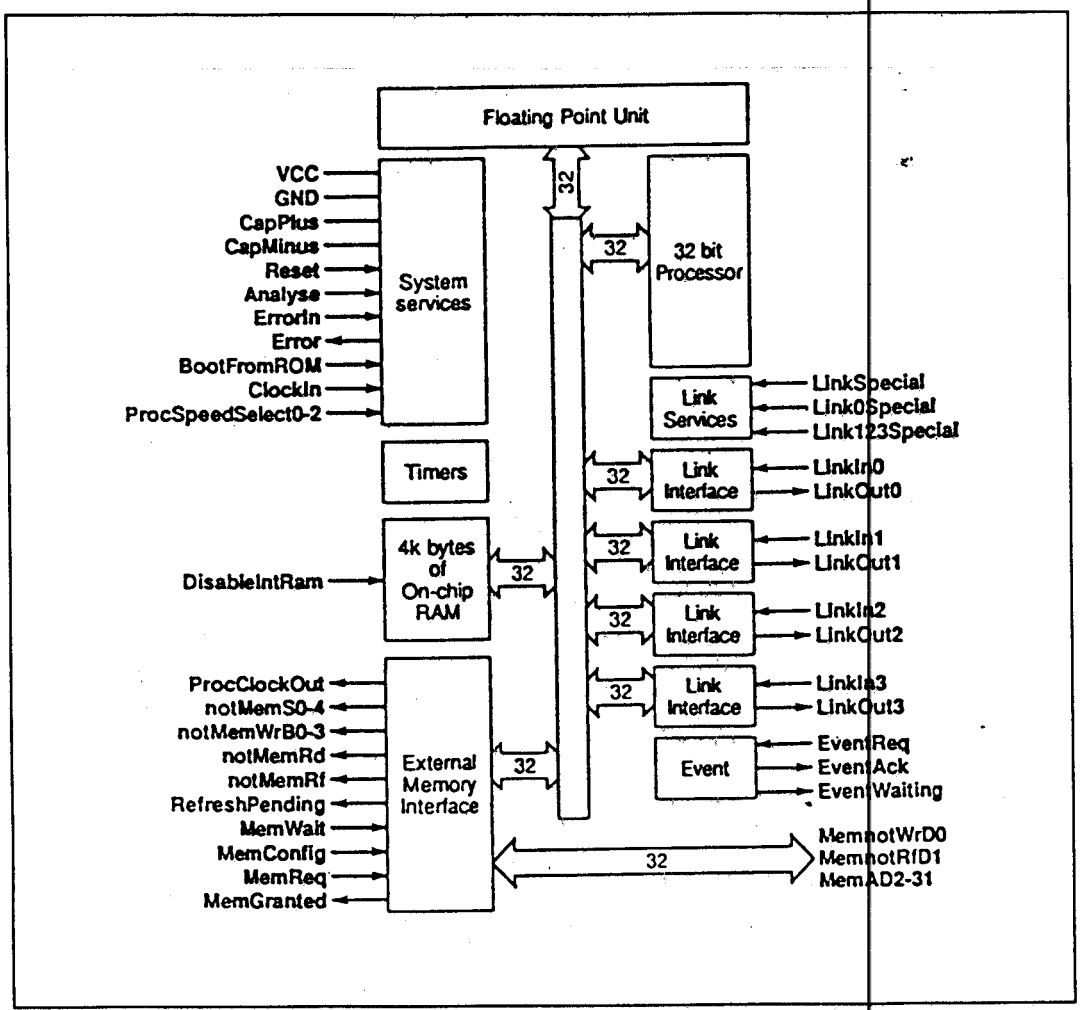
Transputer yang dipakai dalam tugas akhir ini adalah T805. T805

---

[23]----, *Transputer Architecture and Overview*, Computer System Architects, USA, 1990, hal. 6.

[24]ibid., hal. 7.

mempunyai fasilitas standar yang dimiliki oleh keluarga transputer : Prosesor integer (32-bit), link komunikasi (4 buah), RAM internal (4 Kbyte), External Memory Interface (EMI), Event dan Timer. Dan pada tipe ini ada tambahan Floating Point Unit (FPU) 64-bit. Gambar 3-9 memperlihatkan diagram blok dari transputer T805.



Gambar 3-9
Diagram Blok Transputer T805[25]

[25]----, *Transputer Technical Specifications*, op.cit., hal. 3.

3.6.1 Prosessor

Prosessor integer 32-bit berisi instruksi proses logika, pointer instruksi dan proses, dan sebuah register operand. Prosessor mengakses secara langsung 4 Kbyte memori internal kecepatan tinggi, yang dapat digunakan untuk menyimpan data atau program. Bila jumlah memori yang dibutuhkan besar maka prosessor dapat mengakses sampai 4 Gbyte memori melalui Interface Memori Eksternal (Eksternal Memory Interface EMI).

3.6.2 Floating Point Unit

Floating Point Unit (FPU) 64-bit menyediakan perhitungan aritmatik *floating point* 32-bit dan 64-bit. Dengan adanya FPU, memungkinkan operasi aritmatik FP secara konkuren dengan prosessor (CPU), dengan kecepatan 3,3 Mflops pada clock 30 MHz. Semua komunikasi data antara memori dan FPU dibawah kendali CPU.

FPU terdiri dari penghitung terkodemikro dengan 3 register evaluasi floating point untuk manipulasi bilangan-bilangan desimal. Register-register ini yakni FA, FB dan FC, masing-masing dapat menerima 32 bit atau 64 bit data. Kelakuan dari register FPU seperti pada register prosesor integer (CPU).

Walaupun CPU dan FPU dapat bekerja secara kunkuren, tapi ada saat dimana yang satu harus menunggu yang lain. Titik pada sederetan instruksi dimana dibutuhkan transfer data dari atau ke FPU disebut titik sinkronisasi. Pada titik sinkronisasi, unit pemrosesan yang selesai lebih dulu (CPU atau FPU) harus menunggu yang lain selesai. Untuk kemudian dilakukan transfer data dan kedua

prosesor akan berjalan lagi secara konkuren. Untuk memanfaatkan konkurensi secara maksimal, alamat-alamat asal *(source)* dan tujuan *(destination)* dari data floatng dapat dihitung oleh CPU sementara FPU melakukan operasi pada data-data sebelumnya.

3.6.3 Memori

Transputer IMS T805 mempunyai 4 Kbytes memori statik internal kecepatan tinggi untuk meletakkan data. Setiap akses ke memori internal membutuhkan satu cycle prosesor *ProcClockOut*. Transputer juga dapat mengakses eksternal memori sebesar 4 GByte. Memori internal dan memori eksternal adalah bagian dari ruang alamat linier yang sama. RAM Internal dapat di-non-aktifkan dengan memberi logika tinggi pada pin *DisableIntRam* . Setelah itu semua alamat-alamat internal akan dipetakan kepada memori eksternal.

Memori internal dimulai dari alamat paling negatif #80000000 sampai #80000FFF (untuk tipe T805). Memori yang boleh digunakan oleh pemakai dimulai dari #80000070; lokasi ini dinamakan *MemStart*. Instruksi *idmemstartval* digunakan untuk mendapatkan harga dari *MemStart*.

Memori eksternal dimulai dari #80001000 melewati #00000000 sampai dengan #7FFFFFFF. Data konfigurasi memori dan instruksi bootstrap ROM harus dalam ruang alamat paling positif, yaitu masing-masing dimulai pada #7FFFFF6C dan #7FFFFFFE. Ruang alamat tepat dibawah daerah ini biasanya digunakan untuk instruksi-instruksi lain dalam ROM.

3.6.4 Timer

T805 mempunyai timer 32-bit yang akan ber-'detak' secara periodik. Timer memberikan pewaktuan proses yang akurat, memungkinkan suatu proses untuk di-*deshedule* beberapa saat.

Transputer mempunyai dua timer, untuk prioritas rendah dan prioritas tinggi. Timer untuk prioritas tinggi berdetak setiap 1 mikrodetik. Sedangkan timer untuk prioritas rendah berdetak setiap 64 mikrodetik, memberikan 15625 detak setiap detik.

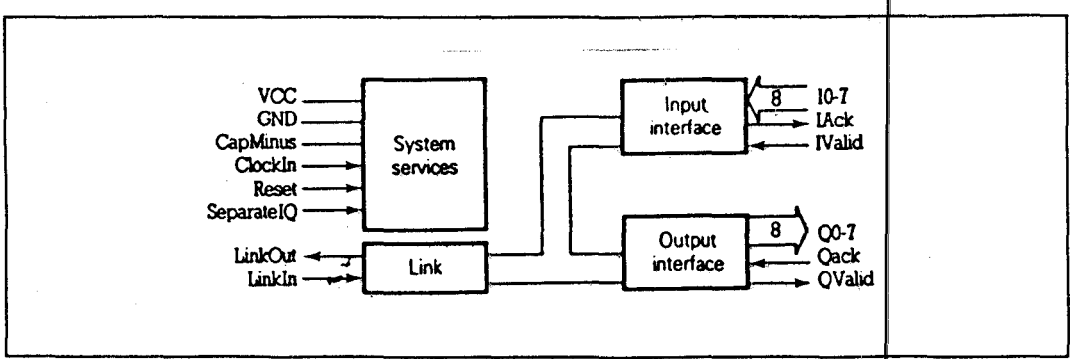
3.7 Link Adaptor

Link adaptor adalah komponen interface yang digunakan oleh transputer untuk berhubungan dengan periferal ataupun bus. Komunikasi link ini merupakan komunikasi *full duplex* dan dilakukan dengan cara mengkonversi data serial ke data paralel ataupun sebaliknya *(bi-directional)*.

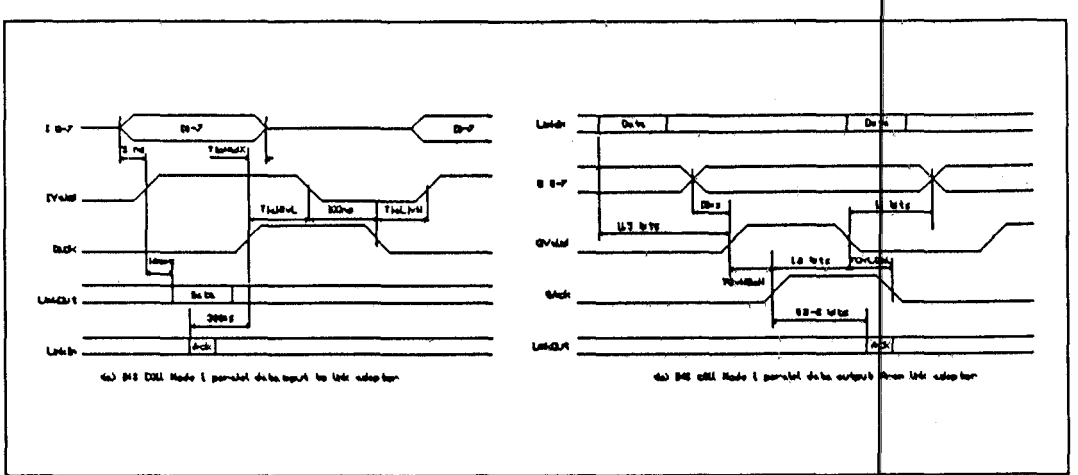
Dalam keluarga transputer INMOS ada dua tipe komponen link adaptor untuk dua tujuan. IMS C011 adalah adaptor untuk interface periferal dan IMS C012 untuk interface bus. Dengan adanya link adaptor , link dapat digunakan sebagai *general purpose I/O*. IMS C011 mempunyai dua mode operasi : mode 1 memberikan terminal input dan output 8 bit yang terpisah; sedangkan mode 2 memberikan terminal input dan output dua arah yang digunakan untuk interface dengan bus dari suatu sistem mikroprosesor. Mode 2 ini merupakan fungsi yang identik dengan C012.

3.7.1 Operasi Mode 1 C011

Gambar 3-10 memperlihatkan diagram blok dari C011 yang beroperasi pada mode 1. Pada mode ini, C011 digunakan sebagai interface periferal dengan terminal input dan output terpisah.



Gambar 3-10
Diagram blok C011 mode 1[26]



Gambar 3-11
Diagram waktu input dan output data C011 mode 1[27]

Gambar 3-11(a) memperlihatkan diagram waktu dari *handshaking* untuk

---

[26]----, *Transputer Handbook*, op.cit., hal. 114.

[27]----, Inmos IMS C011 Link Adaptor Engineering Data.

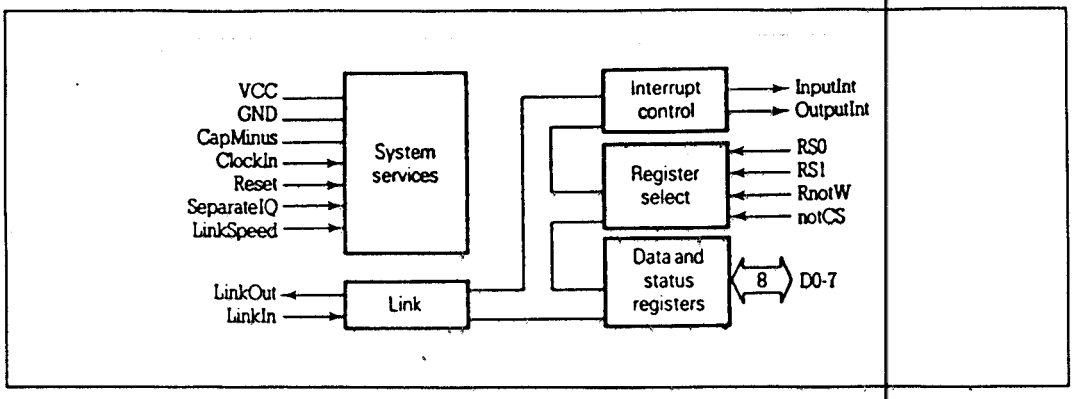
input data ke link adaptor. Data di-*set-up* pada input I0-I7. Setelah waktu minimum untuk *set-up* data, periferal memulai handshaking dengan memberi logika tinggi pada **IValid**. Kemudian link adaptor mengirimkan data ini melalui serial link, dan menunggu *acknowledge* yang menandakan bahwa data sudah diterima. Jika *acknowledge* sudah diterima melalui link input, link adaptor akan menge-set **IAck** ke logika tinggi. Selanjutnya periferal mengembalikan **IValid** ke logika rendah dan link adaptor mengembalikan **IAck** ke logika rendah untuk menyelesaikan handshaking.

Gambar 3-11(b) memperlihatkan diagram waktu dari handshaking untuk output data dari link adaptor. Data yang diterima pada link diletakkan pada output Q0-Q7, dan **QValid** diberi logika tinggi untuk menandakan bahwa data output telah siap. Jika data telah diterima, periferal akan memberi logika tinggi pada **QAck**. Kemudian link adaptor mengirimkan *acknowledge* ke link dan mengembalikan **QValid** ke logika rendah.

Delay minimum antara sisi naik pada **QValid** dan **QAck** adalah nol, demikian pula pada sisi turunnya. Oleh karena itu keduanya (**QValid** dan **IAck**) dapat dihubungkan secara langsung. Ini memberikan handshaking secara otomatis; artinya setiap ada data yang diterima dari link, akan dikeluarkan ke Q0-Q7 dan langsung di-*acknowledge*.

3.7.2 Operasi Mode 2 C011

Gambar 3-12 memperlihatkan diagram blok C011 yang beroperasi pada mode 2.



Gambar 3-12
Diagram blok C011 mode 2[28]

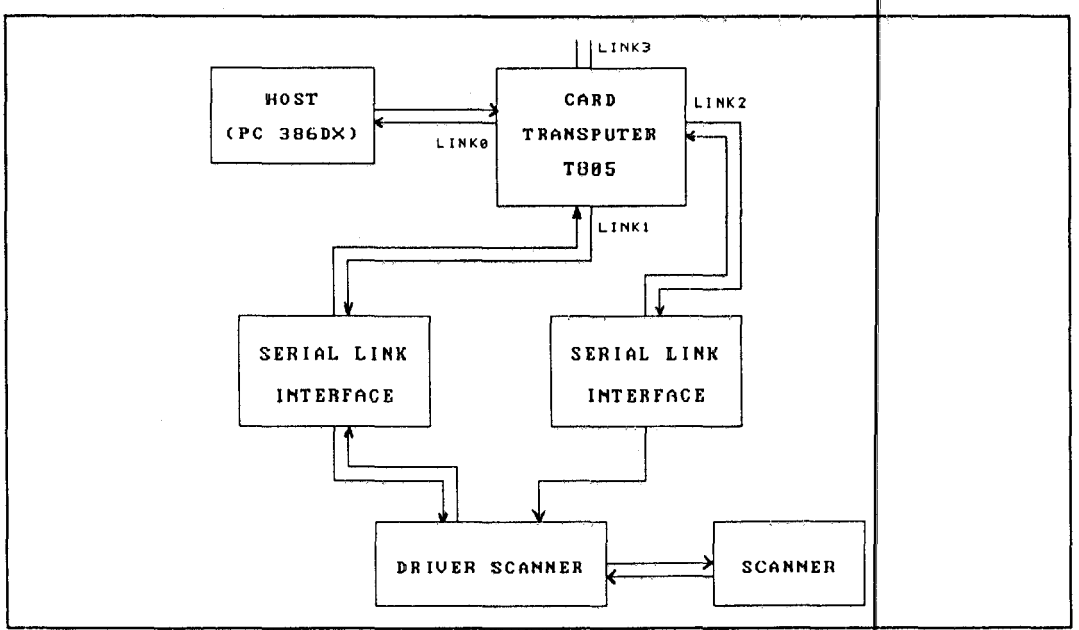
Pada mode ini C011 digunakan sebagai interface antara link dengan bus mikroprosesor, dan biasanya didekoding ke dalam address space seperti layaknya periferal. Register internal yang mengontrol status input dan output dipilih menggunakan **RS0** dan **RS1** bersama-sama dengan **RnotW**. Sinyal-sinyal interupsi dibangkitkan pada **InputInt** dan **OutputInt** untuk menandakan bahwa data yang diterima telah ada, atau bahwa output siap menerima data.

[28]----, *Transputer Handbook*, loc.cit.

BAB IV

PERENCANAAN PERANGKAT KERAS

Perangkat keras yang dibuat merupakan bagian dari pemrosesan awal. Gambar 4-1 memperlihatkan diagram hubungan perangkat keras yang dipakai.



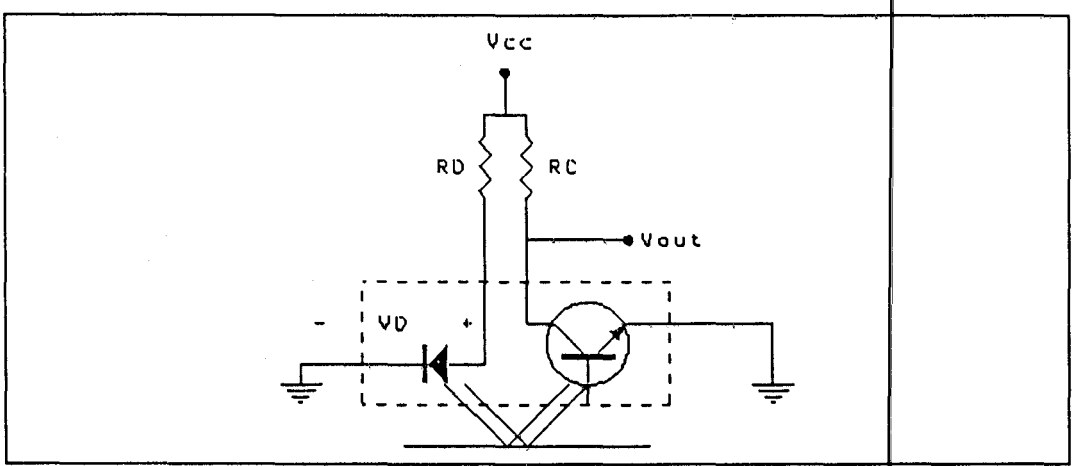
Gambar 4-1
Diagram blok sistem

Pada tugas akhir ini yang direncanakan dan dibuat oleh penulis adalah blok scanner dan drivernya serta serial link interface.

4.1 Sensor

Sensor yang digunakan adalah fotoreflektor. Fotoreflektor ini terdiri atas 2 bagian, yakni dioda sebagai pemancar dan transistor sebagai penerima. Pancaran

dari dioda dipantulkan oleh bidang pantul (kertas). Pada bidang pantul terang atau putih, sebagian besar pancaran diterima oleh transistor. Pada keadaan ini konduktivitas transistor naik, arus kolektor naik, sehingga tegangan jatuh pada R_C (V_{RC}) naik dan V_{out} rendah. Sebaliknya pada bidang pantul gelap atau hitam, konduktivitas turun sehingga V_{out} naik.



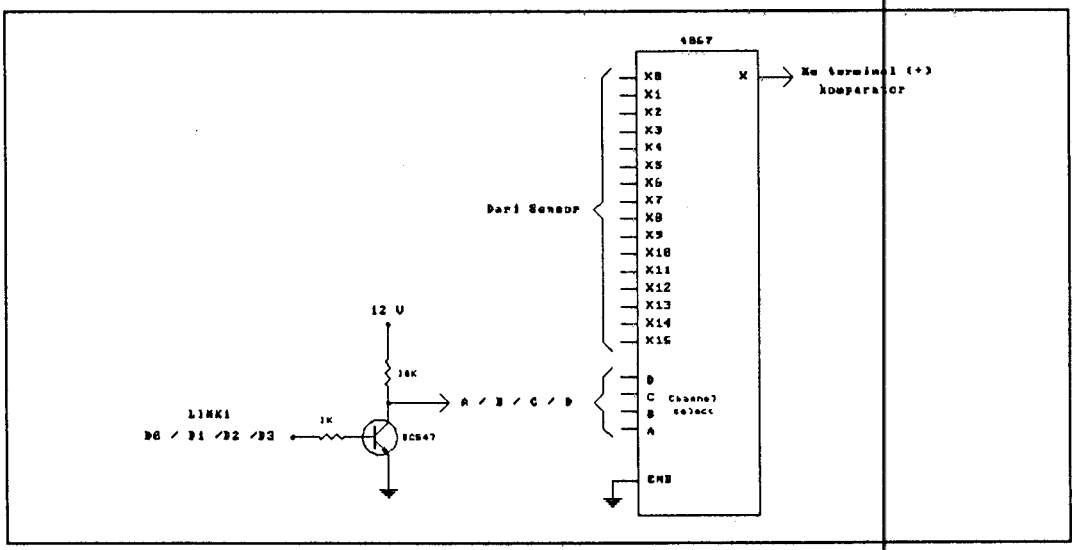
Gambar 4-2
Rangkaian fotoreflektor

Arus steady state normal dari dioda (I_F) menurut data sheet adalah 50 mA. Dalam perencanaan, penentuan besar arus ini dipengaruhi oleh jarak sensor dari kertas dan luas daerah pemancaran. Dipilih R_D = 390 Ohm, sehingga diperoleh : $I_F = (V_{CC} - V_D) / R_D$. Dari percobaan dapat diketahui V_D = 1,2 Volt. Maka $I_F = (12 - 1,2) / 390 = 27,7$ mA.

Arus yang timbul pada transistor (I_C) mempunyai orde ratusan mikroAmpere. Sedangkan saat terdeteksi warna hitam arus akan turun hingga puluhan mikroAmpere, yang oleh karenanya tegangan jatuh pada R_C akan turun dan V_{out} naik. Pemilihan R_C dilakukan sedemikian hingga pada saat terdeteksi warna hitam V_{out} berada di sekitar 5 Volt. Untuk itu dipilih R_C sebesar 39 KOhm.

4.2 Multiplekser Analog

Karena sensor (fotoreflektor) yang digunakan sebanyak 16 buah, untuk menentukan sensor yang akan diambil datanya digunakan multiplekser analog. Dalam hal ini dipilih IC CMOS 4067 dengan 16 input termultipleks.



Gambar 4-3
Rangkaian Multiplekser

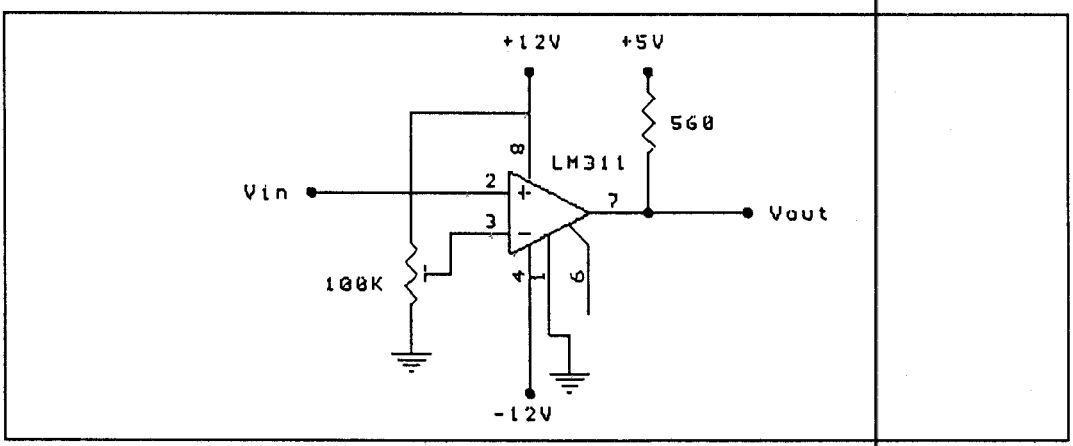
Dipilih V_{DD} sebesar 12 Volt. Tegangan ini masih di bawah V_{DD} maksimum untuk CMOS sebesar 15 Volt. Dengan V_{DD} sebesar ini tegangan analog yang dapat dilewatkan adalah 0 - 12 Volt.

Pada V_{DD} sebesar 12 Volt, tegangan input kontrol level rendahnya (V_{IL}) sebesar 3,5 Volt maksimum, dan tegangan input level tingginya (V_{IH}) sebesar 8,5 Volt minimum. Karena tegangan level rendah maksimum untuk C011 adalah 0,8 Volt dan tegangan level tingginya maksimum adalah 2,4 Volt, maka diperlukan level translator ke dalam daerah tegangan CMOS di atas. Untuk itu ditambahkan transistor (BC 547) dan resistor pull-up pada setiap input kontrol.

4.3 Komparator

Pada saat sensor mendeteksi adanya warna hitam, output komparator akan berlogika tinggi. Komparator bertugas untuk mendeteksi apakah tegangan output melewati ambang batas.

Untuk rangkaian komparator digunakan IC LM311 yang merupakan komparator presisi. IC dipilih karena kefleksibelannya. Outputnya tidaklah $\pm$Vsat seperti komparator biasa, tetapi dapat dipilih dengan jalan mem-pull-up outputnya ke tegangan yang diinginkan. Resistor pull-up dipilih sebesar 560 Ohm.



Gambar 4-4
Rangkaian Komparator.

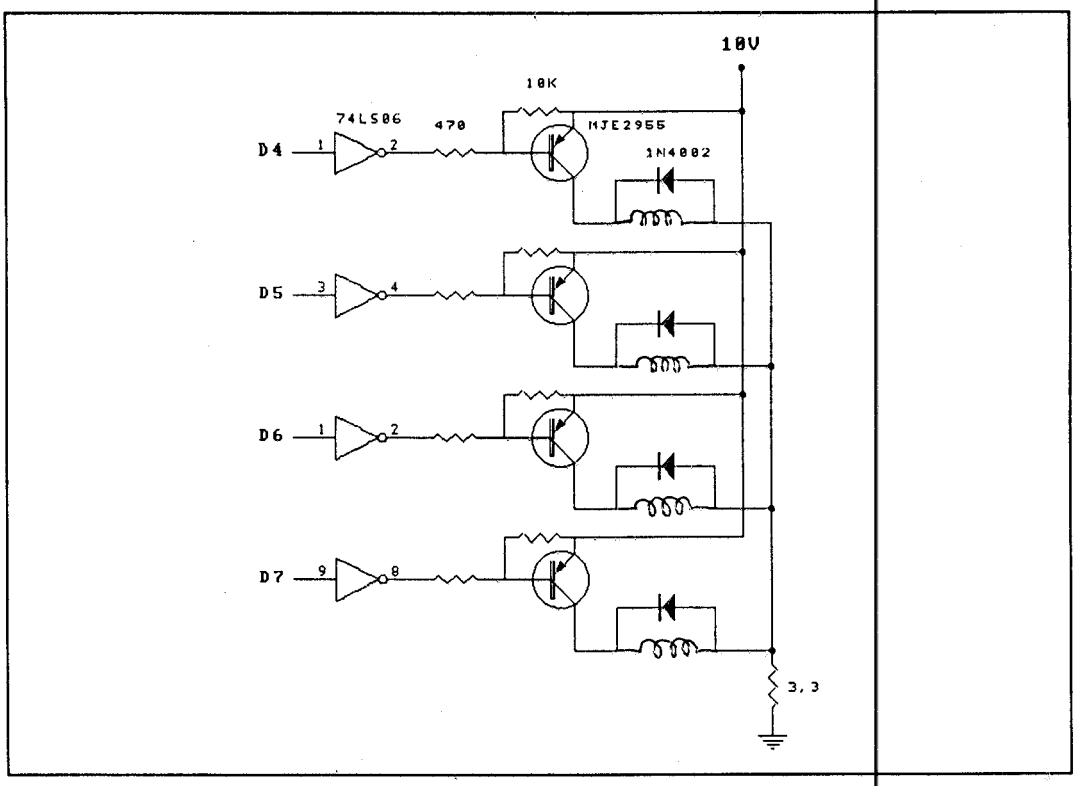
Tegangan referensi diatur berdasarkan tebal garis yang paling tipis yang dapat dideteksi oleh fotoreflektor.

4.4 Driver Motor Stepper

Sensor dapat digerakkan dalam arah sumbu X dan sumbu Y. Untuk itu digunakan dua motor stepper. Motor stepper yang digunakan mempunyai empat koneksi kumparan medan atau lebih dikenal dengan motor stepper 4 phase.

Rangkaian driver untuk motor empat phase ini lebih sederhana.

Gambar 4-5 memperlihatkan rangkaian driver motor stepper yang dipakai sebagai penggerak sensor dalam arah X.



Gambar 4-5
Rangkaian driver motor stepper x.

Pada rangkaian tersebut terdapat dioda clamp untuk membatasi tegangan pada kumparan medan dari motor dan mencegah rusaknya transistor akibat tegangan induksi yang terlalu pada saat transistor tidak menghantar.

Motor stepper yang digunakan mempunyai nilai nominal V_L = 6,8 Volt, I = 0,75 Ampere. Untuk driver digunakan transistor MJE2955. Untuk menambah torsi motor, diberikan tegangan catu sebesar 10 Volt dan ditambahkan resistor R

yang besarnya dapat dihitung sebagai berikut :

$$R = (V_{CC} - V_{CE} - V_L)/I$$

$$R = (10 - 1 - 6,8)/0,75 = 2,2/0,75 = 2,93 \text{ Ohm}$$

Dipilih harga R sebesar 3,3 Ohm.

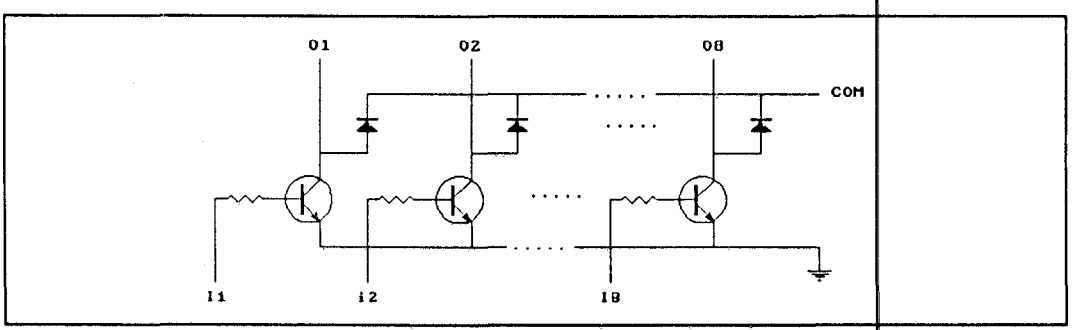
Motor stepper yang kedua digunakan untuk menggerakkan sensor dalam arah Y mempunyai data sebagai berikut : V_L = 8,4 Volt dan I = 0,12 Ampere. Untuk driver digunakan IC ULN2803 yang berisi delapan transistor driver. IC ini cukup kuat untuk menangani arus yang dibutuhkan motor stepper. Untuk menambah torsi motor, diberikan catu tegangan sebesar 10 Volt dan ditambahkan resitor R yang besarnya dapat dihitung sebagai berikut :

$$R = (V_{CC} - V_{CE} - V_L)/I$$

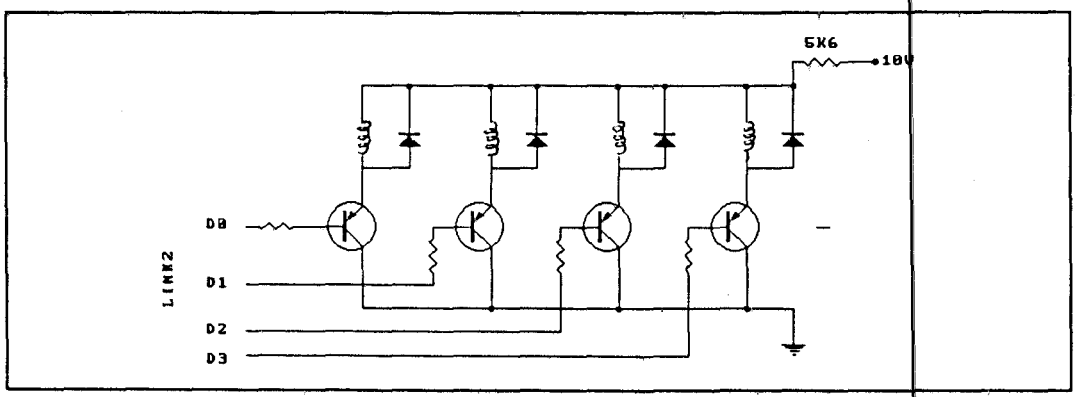
$$R = (10 - 1 - 8,4)/0,12$$

$$R = 0,6/0,12 = 5 \text{ Ohm}$$

Untuk itu dipilih harga R = 5,6 Ohm.



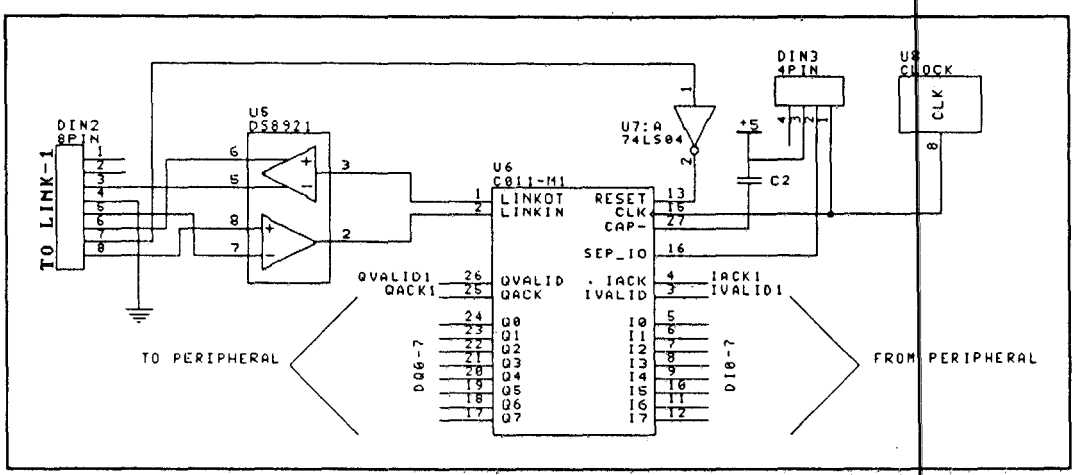
Gambar 4-6
Diagram koneksi ULN2803



Gambar 4-7
Driver motor stepper y.

4.5 Serial Link Interface

Untuk rangkaian serial link interface digunakan IC dari keluarga transputer INMOS, yaitu IMS C011. IC ini dapat dioperasikan dalam dua mode, mode 1 dan mode 2.Dalam tugas akhir ini dipakai mode 1. Mode 1 dipilih dengan jalan menghubungkan pin **SeparateIQ** ke V_{CC} (untuk keperluan 10 Mbit/dtk) atau ke Clockin (untuk kecepatan 20 Mbit/dtk).



Gambar 4-8
Serial Link Interface

Pin Clockin dihubungkan osilator dengan frekuensi standar untuk komponen-komponen keluarga transputer, yaitu 5 Mhz. Kestabilan osilator ini sangat penting, untuk itu digunakan osilator kristal. Frekuensi osilator eksternal ini akan dikalikan oleh Phase Lock Loop (PLL) on-chip untuk menghasilkan clock internal. Ini membutuhkan kapasitor dekopling sebesar 1μF yang dihubungkan antara pin V_{CC} dan **CapMinus**. Kapasitor dekopling ini harus berkualitas tinggi dan sesuai untuk frekuensi tinggi, untuk itu digunakan kapasitor tantalum.

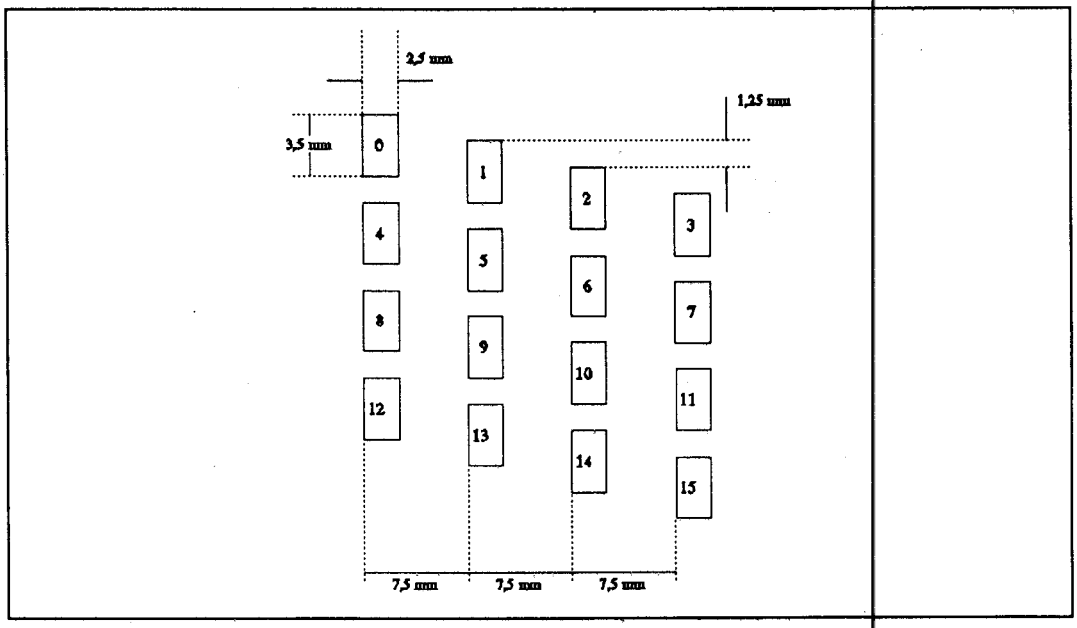
Pin **LinkOut** dan **LinkIn** masing-masing dihubungkan dengan differential driver and receiver yang dikemas dalam IC DS8921. Differential Driver and Receiver ini memberikan kekebalan yang maksimum terhadap noise.

4.6 Perencanaan Mekanik

4.6.1 Tata Letak Sensor

Kepala scan yang dirancang terdiri atas 16 fotoreflektor; jadi dalam satu karakter ketinggian dari karakter adalah sebanding dengan 16 pengambilan data. Dengan pertimbangan meminimkan ukuran karakter yang akan discan dan karena ukuran fisik fotoreflektor yang cukup besar (tinggi 3,5 mm dan lebar 2,5 mm), maka peletakan fotoreflektor dilakukan dengan cara khusus. Keenambelas fotoreflektor diatur dalam 4 kolom (masing-masing kolom terdiri atas 4 fotoreflektor). Jarak antar kolom adalah 7,5 mm. Sedangkan posisi horisontal dari kolom pertama dengan kolom kedua mempunyai beda tinggi sebesar 1,25 mm. Dengan demikian jika data pertama dari satu kolom pada karakter diambil oleh

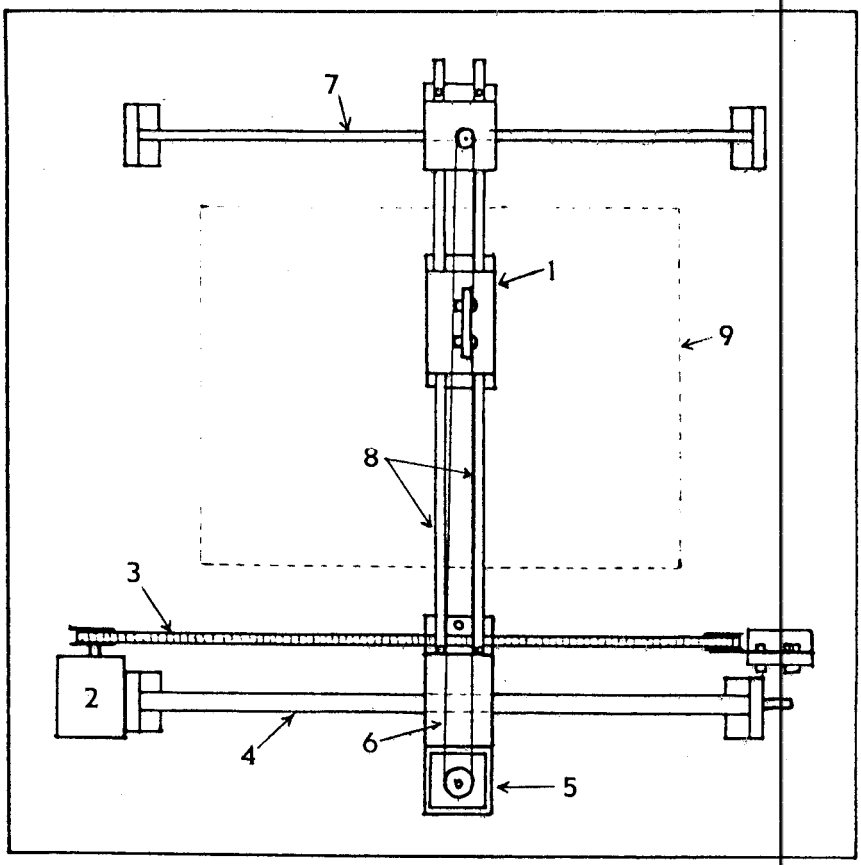
fotoreflektor teratas dari kolom pertama pada kepala scan, maka data kedua akan diambil oleh fotoreflektor teratas dari kolom kedua pada kepala scan. Jadi, tinggi karakter maksimum yang dibaca sekitar 20 mm.



Gambar 4-9
Tata letak fotoreflektor.

4.6.2 Konstruksi Mekanik

Scanner yang dirancang mempunyai dua arah gerakan kepala scan, yaitu arah X dan arah Y. Pada saat scanning, yang bekerja hanyalah gerakan dalam arah X, sedangkan gerakan arah Y digunakan untuk menentukan letak baris yang akan di-scan. Gambar 4-10 memperlihatkan tampak atas dari konstruksi mekanik scanner.



1. Kepala Scan
2. Motor Stepper X
3. Belt X
4. Penumpu X1
5. Motor Stepper Y
6. Belt Y
7. Penumpu X2
8. Penumpu Y
9. Bidang Scan

Gambar 4-10
Tampak atas konstruksi mekanik scanner.

BAB V

PERENCANAAN PERANGKAT LUNAK

5.1 Pendahuluan

Perangkat lunak dalam tugas akhir ini terbagi dalam dua bagian, yakni yang berada di PC (host) dan di transputer. Perangkat lunak di transputer merupakan inti dari sistem, sedangkan perangkat lunak di PC bertugas melayani kebutuhan transputer baik input keyboard, akses ke disket, maupun tampilan ke monitor.

Perangkat lunak di transputer pada prinsipnya terbagi menjadi 3 tugas, yaitu pengambilan data, pengolahan awal dan jaringan saraf tiruan. Perangkat lunak jaringan yang direncanakan adalah Error Back-Propagation, yang terdiri atas proses pelatihan (*training*) dan pelaksanaan atau pemakaian jaringan.

Perangkat lunak di transputer menggunakan compiler Logical Systems C versi 89.1 yang memang dikembangkan untuk transputer, sedangkan untuk di PC digunakan compiler Borland C.

5.2 Aspek-aspek dalam perangkat lunak transputer

Dalam merancang perangkat lunak untuk transputer pada dasarnya sama dengan perancangan pada PC. Hanya saja ada beberapa aspek yang perlu diperhatikan agar tidak terjadi hal-hal yang tidak diinginkan. Termasuk dalam aspek-aspek ini yaitu : pengalokasian memori dinamik dan transfer data antar

proses maupun transfer data dengan host (PC).

5.2.1 Alokasi memori dinamik

Untuk merencanakan perangkat lunak jaringan saraf tiruan yang perlu diperhatikan adalah masalah alokasi memori. Kemudahan alokasi memori dalam jumlah besar untuk penyimpanan bobot sementara dan parameter-parameter selama proses pelatihan menunjukkan kelebihan transputer. Hal ini tidak seperti DOS yang membatasi alokasi memori maksimum 64 KB untuk sekali alokasi. Transputer memungkinkan untuk alokasi sebanyak yang diinginkan tergantung pada jumlah memori yang ada. Akan tetapi rutin yang tersedia (*malloc()*) hanya mengijinkan alokasi maksimum 128 KB, baik untuk sekali alokasi ataupun total dari bebarapa kali alokasi. Untuk mengatasi masalah ini, maka dibuatlah rutin sendiri. Rutin ini merupakan rutin alokasi memori yang pengalokasiannya dilakukan di ruang memori di atas *heap* asli dari program.

Pada awal program ditentukan terlebih dahulu awal dari *heap* yang baru (*my_heapstart*), yaitu diatas dari *heap* yang asli - heap yang asli berada pada 128 KB pertama di atas kode program. Kemudian meletakkan pointer dari ujung heap pada *my_heaptop*. Pointer *my_heaptop* menunjukkan posisi awal dari setiap kali alokasi memori.

```
my_heapstart        = heapstart + 128*1024;
my_heaptop          = my_heapstart;
```

5.2.2 Transfer data antar proses

Dalam tingkat perangkat lunak, transfer data antar proses dalam transputer maupun transfer data dengan host dilakukan melalui *channel*. Komunikasi ini harus sinkron, artinya penentuan jumlah byte data yang akan diterima oleh suatu proses harus sama dengan jumlah byte yang dikirim oleh proses pengirim. Kesalahan dalam komunikasi ini dapat mengakibatkan dua kemungkinan, yang pertama transfer data terhenti (*deadlock*) karena data yang harus diterima kurang atau ada sisa data yang tidak bisa dikirim, kedua adanya kekeliruan data yang diterima.

5.3 Pemrosesan Awal

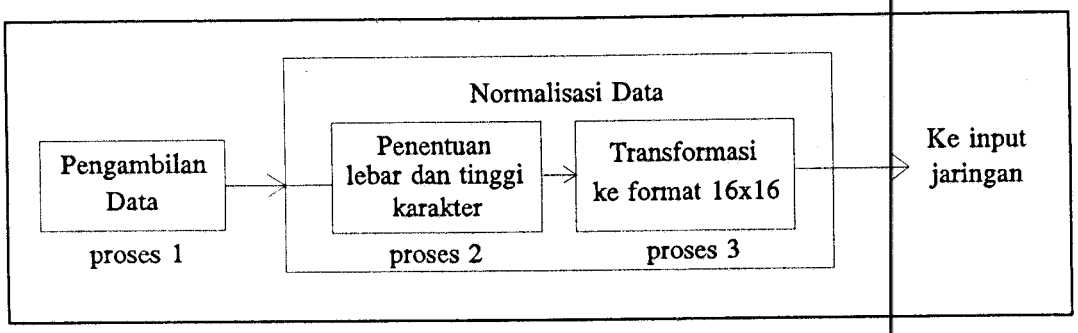
Sebelum diumpankan ke jaringan saraf tiruan, serangkaian input harus diproses terlebih dahulu. Proses ini dikenal sebagai pemrosesan awal (*pre-processing*). Pemrosesan awal ditujukan agar format input sesuai dengan format input jaringan. Selain itu pemrosesan awal yang baik akan menambah performa sistem.

Perangkat keras yang telah dirancang merupakan bagian dari pemrosesan awal. Sedangkan perangkat lunak yang direncanakan terdiri atas perangkat lunak untuk pemrosesan awal dan perangkat lunak jaringan saraf tiruan.

Pemrosesan awal dimulai dengan pengambilan data dan dimasukkan ke buffer. Dari buffer, data diambil per karakter untuk dicari lebar dan tingginya, kemudian dikonversi ke format 16x16 yang merupakan format input jaringan.

Gambar 5-1 memperlihatkan diagram dari perangkat lunak untuk

pemrosesan awal.



Gambar 5-1
Urutan proses pengolahan data

5.3.1 Pengambilan data

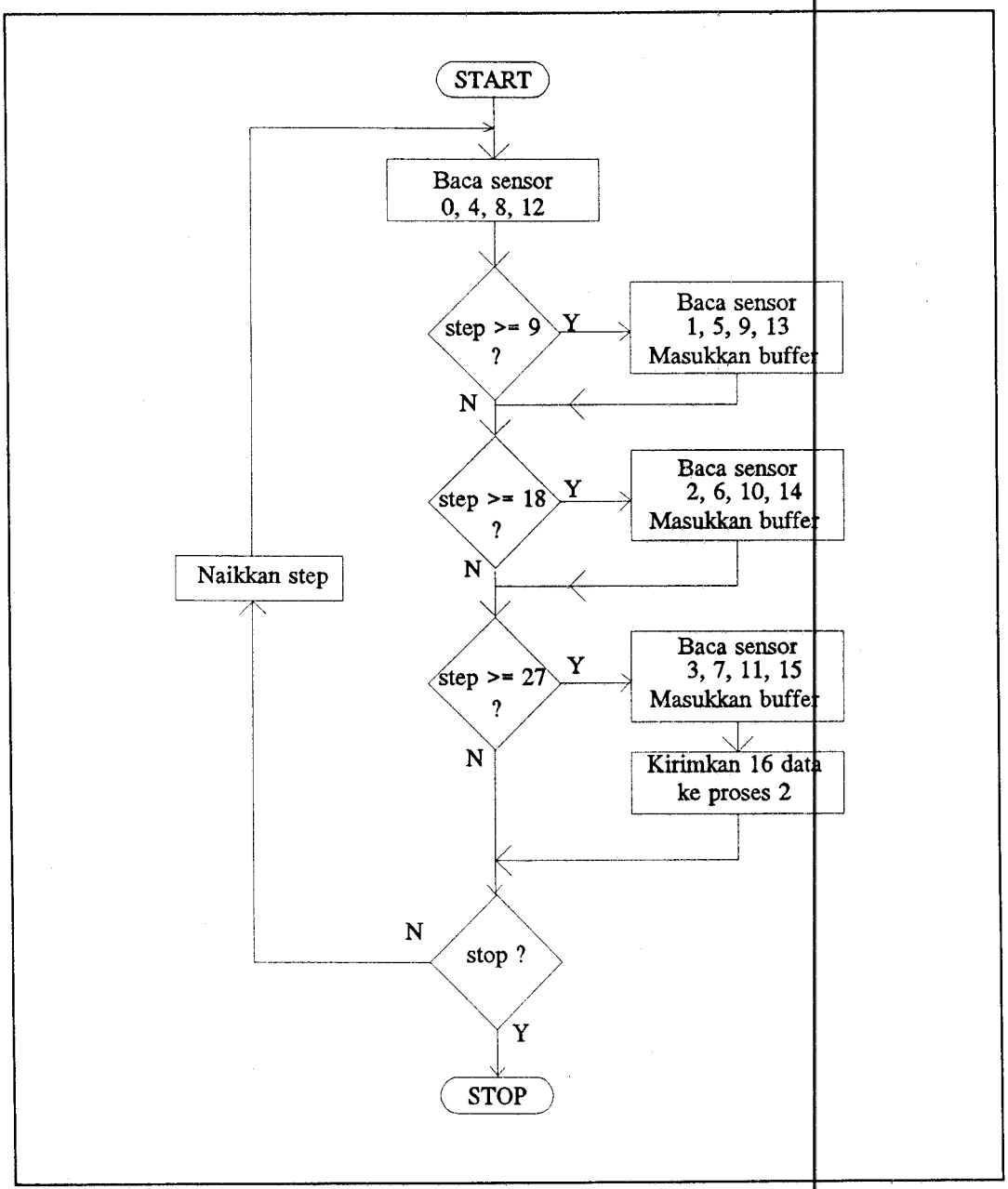
Untuk merencanakan algoritma pengambilan harus meninjau kembali Gambar 4-9 yang memperlihatkan tata letak dari sensor. Juga perlu dihitung dahulu gerakan linier putaran motor stepper. Motor stepper X mempunyai spesifikasi 1,8 derajat per step dan jari-jari diameter roda gigi 13,5 mm. Gerakan linier dari motor stepper X dapat dihitung sebagai berikut :

$$x = \pi/180 \times 1{,}8^{\circ} \times 13{,}5 \text{ mm}$$

$$x = 0{,}424 \text{ mm}$$

Jadi, satu step memberikan gerak linier sepanjang 0,424 mm. Jarak antar kolom pada kepala sensor adalah 7,5 mm, atau 7,5/0,424 = 18 step. Pengambilan data dilakukan setiap dua step atau 0,848 mm, atau disini digunakan istilah 1 step-sensor = 2 step motor. Dengan demikian dapat dibuat algoritma sebagai berikut: Jika untuk pengambilan data pertama pada kolom pertama dari karakter, dilakukan pada step-sensor ke 0 oleh sensor teratas pada kolom sensor pertama (sensor 0), maka untuk data kedua akan diambil oleh sensor teratas pada kolom sensor kedua (sensor 1) pada step-sensor ke-9, dan pada step-sensor ke-18 untuk data

berikutnya. Demikian seterusnya hingga keenambelas data untuk satu kolom karakter terlengkapi. Gambar 5-2 memperlihatkan diagram alir dari proses pengambilan data.



Gambar 5-2
Diagram alir proses pengambilan data

5.3.2 Penentuan lebar dan tinggi karakter

Pertama kali yang harus dilakukan adalah mencari lebar dari karakter yang dibaca. Proses ini mengambil satu per satu 16 data pertama kolom pertama, dan dicek apakah ada data yang tidak sama dengan nol. Jika belum ada data yang tidak sama dengan nol pengecekan dilanjutkan ke kolom berikutnya. Jika sudah ada satu data yang lebih dari nol pada kolom tersebut indeks kolom (batas kanan) diberi nilai nol.

Pengecekan dilanjutkan ke kolom-kolom berikutnya sampai ditemukan semua data pada satu kolom bernilai nol. Pada kolom sebelum kolom tersebut merupakan batas kiri dari karakter. Dengan demikian diperoleh lebar dari karakter.

Setelah didapatkan batas kiri dan batas kanan dari karakter, langkah selanjutnya adalah mencari batas atas dan batas bawah untuk menentukan tinggi karakter. Pengecekan dimulai dari baris paling atas; apakah ada data yang tidak nol. Jika tidak ada, dilanjutkan ke baris berikutnya sampai ditemukan data yang tidak nol. Baris ini merupakan batas atas dari karakter. Untuk mencari batas bawah dilakukan langkah yang sama, tetapi pengecekan dimulai dari bawah.

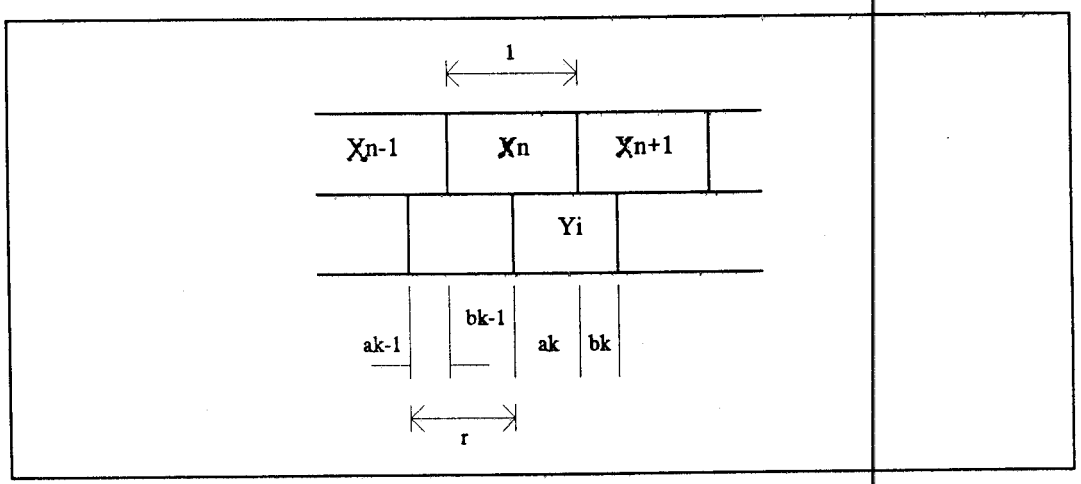
Langkah berikutnya adalah mengirimkan lebar dan tinggi karakter beserta data-data yang ada dalam batas-batas tersebut kepada proses yang mengkonversi ke format 16 x 16.

5.3.3 Transformasi Data ke Format 16 x 16

Jaringan saraf tiruan Error Back-Propagation yang dirancang untuk

mengenali pola angka ini mempunyai input sebanyak 256 (16 x 16). Jumlah input ini dipilih berdasarkan pertimbangan, dengan input sebanyak ini telah mempunyai resolusi yang memadai untuk membedakan 10 angka. Tetapi mungkin saja jumlah ini terlalu sedikit jika diterapkan pada huruf atau karakter yang lebih kompleks.

Transformasi format data ditujukan untuk mengubah data yang ada sehingga memenuhi format 16 x 16. Pertama kali yang dilakukan adalah transformasi tinggi karakter. Tinggi karakter maksimum adalah sesuai dengan banyaknya sensor, yakni 16 data. Jika karakter yang di-scan mempunyai tinggi kurang dari 16, atau rasionya (r) kurang dari satu, maka yang harus dilakukan adalah ekspansi data.



Gambar 5-3
Ekspansi Data

x adalah vektor hasil scan suatu karakter untuk satu kolom yang banyaknya sama dengan tinggi karakter. **y** adalah vektor yang diperoleh setelah transformasi, dan mempunyai elemen sebanyak 16. Dianggap panjang tempat lokasi untuk x maupun y pada Gambar 5-3 adalah sama. Sehingga jika panjang satu elemen x adalah 1, maka panjang elemen y adalah r ($r < 1$). Jika jumlah elemen $\mathbf{x} = n_x$, maka :

$r = n_x/16$

untuk $n_x = 16$ ---> $y = x$

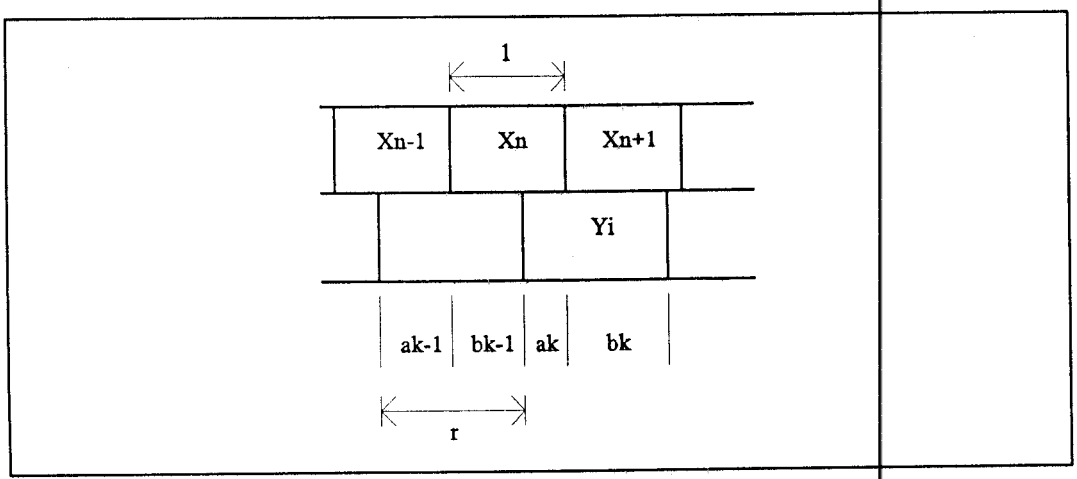
untuk $n_x < 16$ ---> $y_i = a_k x_n + b_k x_{n+1}$

$n = n + 1$

dimana : $a_k = 1 - b_{k-1}$

$b_k = r - a_k$

Sedangkan jika lebar karakter yang di-scan mempunyai dua kemungkinan, lebih dari 16 atau kurang dari 16. Jika kurang dari 16, algoritma transformasinya sama dengan di atas. Jika lebih dari 16, atau rasionya lebih dari 1, maka harus dilakukan ekstraksi data.



Gambar 5-4
Ekstraksi Data

x adalah vektor hasil scan suatu karakter untuk satu kolom yang banyaknya sama dengan lebar karakter. y adalah vektor yang diperoleh setelah transformasi, dan mempunyai elemen sebanyak 16. Dianggap panjang tempat lokasi untuk x maupun y pada Gambar 5-4 adalah sama. Sehingga jika panjang satu elemen x adalah 1,

maka panjang elemen y adalah r ($r > 1$). Jika jumlah elemen $x = n_x$, maka :

$r = n_x/16$

untuk $n_x = 16$ ---> $y = x$

untuk $n_x < 16$ ---> $y_i = a_k x_n + b_k x_{n+1}$

$n = n + 1$

dimana : $a_k = 1 - b_{k-1}$

$b_k = r - a_k$

Data yang telah ditransformasikan siap untuk dimasukkan input jaringan untuk dikenali, atau disimpan ke dalam file untuk persiapan proses pelatihan.

5.4 Jaringan Error Back-Propagation

Dalam implementasi jaringan saraf tiruan ada beberapa langkah yang dilakukan. Langkah-langkah tersebut yakni : pertama adalah persiapan pelatihan, yang menyiapkan parameter-parameter jaringan serta alokasi memori dinamik; kedua tahap pelatihan; dan yang ketiga tahap pemakaian jaringan.

5.4.1. Persiapan pelatihan

Host (PC) membaca parameter-parameter pelatihan untuk dikirimkan ke program pelatihan yang berada di transputer. Parameter-parameter tersebut yaitu :

- Jumlah semua pola yang akan dipelajari (nPatterns).
- Jumlah iterasi (nIterations).

- Banyaknya titik pada lapisan input/Input layer (nInputNodes).
- Banyaknya titik pada lapisan tengah/hidden layer (nHiddenNodes).
- Banyaknya titik pada lapisan output/output layer (nOutputNodes).
- Konstanta belajar (learning constant, η, eta).
- Konstanta momentum (α, alpha).

Setelah transputer menerima parameter ini, langkah berikutnya adalah mengalokasikan memori untuk peletakan sementara dari :

- Output dari lapisan input (**out0**), lapisan tengah (**out1**), dan lapisan output (**out2**) untuk setiap pola input.
- Delta dari lapisan output (**delta2**) dan tengah (**delta1**) untuk setiap pola input.
- Bobot (**w**) dan perubahan bobot (Δ**w**) yang menghubungkan lapisan input (**w1**) dengan lapisan tengah, dan lapisan tengah dengan lapisan output (**w2**).
- Target dari masing-masing pola input.

Kemudian program membaca bobot-bobot awal dan meletakkannya ke memori yang telah dialokasikan. Bobot-bobot awal ini dibuat acak dan bernilai antara -0,3 sampai dengan 0,3. Langkah terakhir dari tahap persiapan ini adalah membaca semua pola input dan targetnya yang akan dilatihkan.

5.4.2 Pelatihan

Langkah pertama tahap pelatihan ini adalah menghitung output (*aktivasi*) dari lapisan tengah maupun lapisan output. Dapat dirumuskan sebagai berikut :

$$out1_h = f(\sum_i w1_{hi} out0_i) \qquad (5\text{-}1)$$

$$out2_j = f(\sum_h w_{jh} out1_h) \qquad (5\text{-}2)$$

dimana : i = jumlah titik pada lapisan input

h = jumlah titik pada lapisan tengah/hidden

j = jumlah titik pada lapisan output

f() = fungsi sigmoid

f(x) = 1 / (1 + exp(-x))

Langkah berikutnya adalah menghitung δ untuk lapisan output (delta2) dan dilanjutkan dengan penghitungan δ untuk lapisan tengah (delta1).

$$\begin{aligned} delta2_j &= (target_j - out2_j) out2_j (1 - out2_j) \\ delta1_h &= \sum_j (delta2_j\ w_{jh}) out1_h (1 - out1_h) \end{aligned} \qquad (5\text{-}3)$$

Setelah penghitungan delta dari lapisan output dan tengah untuk semua pola selesai, kemudian dilakukan penghitungan koreksi bobot (Δw) sebagai berikut :

$$\Delta w(t) = \eta \sum_{p=1}^{P} (delta2_{pj}\ out1_{ph}) + \alpha \Delta w(t-1) \qquad (5\text{-}4)$$

untuk bobot yang menghubungkan lapisan tengah dengan lapisan output, dan

$$\Delta w(t) = \eta \sum_{p=1}^{P} (delta1_{ph}\ out0_{pi}) + \alpha \Delta w(t-1) \qquad (5\text{-}5)$$

untuk bobot yang menghubungkan lapisan input dengan lapisan tengah, dimana :

$\Delta w(t)$ = koreksi bobot sekarang

$\Delta w(t\text{-}1)$ = koreksi bobot sebelumnya

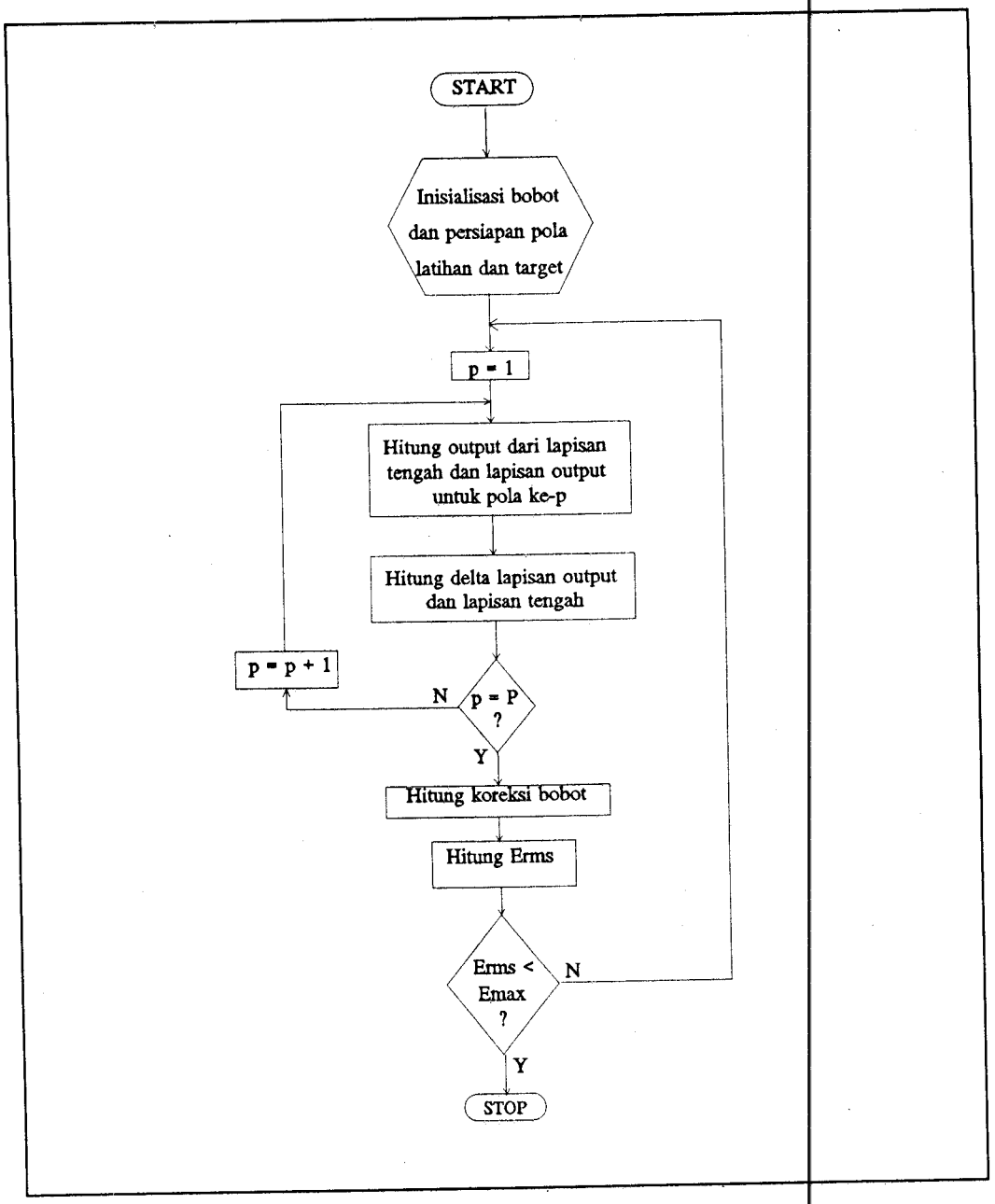
P = jumlah pola input

Koreksi bobot pada suatu saat disimpan untuk digunakan lagi pada perhitungan koreksi bobot berikutnya.

Setiap kali seluruh pola dimasukkan dan dihitung aktivasi serta delta dari lapisan-lapisannya, error rata-rata untuk error dari setiap titik output untuk setiap pola dihitung :

$$E = \frac{1}{PJ} \sum_{p=1}^{P} \sum_{j=1}^{J} (target_{pj} - out2_{pj})^2 \qquad (5\text{-}6)$$

Tahap pelatihan berhenti setelah error telah mencapai level error yang diinginkan atau jumlah iterasi yang diinginkan terpenuhi. Nilai bobot akhir serta nilai aktivasi dari masing-masing pola input dikirimkan ke host untuk disimpan dalam bentuk file.



Gambar 5-5
Diagram alir pelatihan EBP

5.4.3 Tahap Pemakaian

Tahap pemakaian ini mempunyai langkah yang sama dengan dengan langkah pertama dari tahap pelatihan, yakni mengumpankan input ke input jaringan dan menghitung aktivasinya (Pers. (5-1) dan (5-2)). Dari aktivasi ini

dapat ditentukan apakah pengenalan telah dilakukan dengan baik. Hanya saja input jaringan bukan lagi dari pembacaan file, melainkan langsung diambil dari pembacaan scanner. Scanner membaca karakter, membuat normalisasinya, mengirimkan hasilnya ke jaringan, dan jaringan menghitung aktivasi outputnya.

BAB VI

PENGUJIAN SISTEM

Untuk mengetahui sampai sejauh mana performa dari sistem yang telah dibuat, diperlukan adanya suatu pengujian. Ada tiga tahap pengujian yang dilakukan, yakni: pengujian perangkat keras pengambilan data (scanner), pengujian jaringan saraf tiruan dan pengujian sistem secara keseluruhan dengan mengaplikasikannya secara langsung.

6.1 Pengujian perangkat keras

6.1.1 Pengujian serial link interface

Link Interface dipasang pada LINK1. Untuk output data secara free running, QAck dan QValid dihubungkan secara langsung. Setelah itu dikirimkan data dengan perintah ChanOutTimeFail() dengan besar data sebesar 1 byte dan batas waktu sebesar 8000 tick (0.5 detik). Jika pada interval waktu tersebut data belum terkirim, berarti terdapat kesalahan pada modul. Pengiriman yang berhasil dapat dites langsung pada pin-pin Q0 - Q7.

6.1.2 Pengujian scanner

Modul serial link interface pertama dihubungkan dengan LINK1 dan modul kedua dengan LINK2. Modul yang pertama digunakan untuk mengendalikan motor stepper X dan seleksi sensor. Sedangkan modul kedua untuk mengendalikan

motor stepper Y.

6.1.2.1 Pengujian gerak X-Y

Kedua motor stepper diberi data yang dirotasi. Setiap step, counter step dinaikkan. Counter ini menunjukkan posisi dari kepala scan terhadap posisi awal. Instruksi untuk kembali ke posisi awal dengan jalan merotasi data dalam arah yang berlawanan dengan jumlah step yang sama. Dalam pengujian ini motor stepper berjalan baik.

6.1.2.2 Pengujian sensor

Pengujian dilakukan untuk mengetahui nilai tegangan dari output rangkaian sensor pada bidang putih dan bidang hitam. Kepala sensor diletakkan pada bidang putih di salah satu sudut bidang scan. Tegangan dari masing-masing fotoreflektor diukur untuk melihat nilai tegangan yang paling tinggi (V_{p1}). Bidang putih diganti bidang hitam dengan posisi kepala tetap. Tegangan masing-masing fotoreflektor diukur lagi. Ditentukan tegangan yang paling rendah pada saat mendeteksi warna hitam (V_{h1}). Hal yang sama dilakukan pada sudut bidang scan yang lain sehingga diperoleh V_{p2} dan V_{h2}. Ditentukan V_p tertinggi dan V_h terendah. Tegangan referensi dari rangkaian komparator diset pada tegangan antara V_p dan V_h (sekitar 7,5 Volt).

6.2 Pengujian Jaringan Saraf Tiruan

Pada bagian ini akan dibahas mengenai pengujian performa dan

karakteristik dari jaringan saraf tiruan yang diaplikasikan untuk pengenalan pola karakter angka tulisan tangan.

Pengujian dilakukan terhadap tiga parameter jaringan, yaitu : konstanta belajar η, konstanta momentum α dan arsitektur jaringan yang dalam hal ini adalah jumlah sel dalam lapisan tengah (*hidden layer*). Ukuran yang dijadikan pedoman dalam pengujian ini adalah hasil perhitungan error serta melihat langsung kebenaran dari keputusan yang diambil oleh jaringan.

6.2.1. Persiapan pengujian

Jaringan Back-Propagation membutuhkan pengawasan dalam proses pelatihannya (*supervised learning*). Maka sebelum proses pelatihan perlu dipersiapkan lebih dahulu pola-pola yang akan dilatihkan. Pola-pola ini terdiri dari pasangan pola-pola input dan pola-pola targetnya.

Pola input latihan ditulis di atas kertas dan di-scan. Hasilnya yang telah dinormalisasi disimpan ke dalam file bersama-sama dengan target masing-masing. Pola yang disiapkan sebanyak 65 pola yang mencakup berbagai variasi dari angka '0' sampai '9'. Gambar 6-1 memperlihatkan pola-pola yang dipakai dalam pelatihan.

Selain pola latihan dipersiapkan juga pola-pola tes sebanyak 29 pola. Pola ini digunakan untuk melihat hasil belajar jika jaringan diberi input yang tidak sama persis dengan pola-pola yang dilatihkan. Gambar 6-2 memperlihatkan pola-pola yang dipakai untuk pengetesan jaringan.

Gambar 6-1a
Pola-pola yang dilatihkan

Gambar 6-1b
Pola-pola yang dilatihkan(lanjutan)

Gambar 6-2
Pola-pola tes

6.2.2. Pengujian awal

Sebelum dilakukan pengujian terhadap berbagai perubahan parameter pelatihan, dilakukan dahulu pengujian awal. Pengujian awal ditujukan untuk mencari harga error standar yang nantinya akan dijadikan batas error maksimal bagi pengujian selanjutnya.

Untuk pengujian awal dipilih nilai $\eta = 0{,}015$ dan $\alpha = 0{,}01$. Harga batas maksimum error ditentukan 0,01. Kemudian dilakukan iterasi sambil memonitor error. Pada iterasi ke-306 perhitungan berhenti, error yang dihasilkan 0,009976 sedangkan error tes 0,030746. Dilihat hasil latihan baik terhadap pola-pola yang dilatihkan maupun hasil tes. Semua pola latihan sudah dikenali dengan cukup baik, walaupun ada beberapa aktivasi pola latihan yang masih belum cukup kuat. Sebagian besar pola tes (27 pola) dari 29 pola sudah dikenali.

Dengan asumsi bahwa error maksimum sebesar 0,01 sudah cukup, error ini dijadikan batas bagi pengujian selanjutnya.

6.2.3. Pengujian Parameter Belajar dan Arsitektur Jaringan

Pengujian karakteristik jaringan yang dilakukan pertama kali adalah dengan perubahan nilai konstanta momentum α dengan nilai konstanta belajar η yang tetap. Kemudian dilakukan lagi untuk nilai konstanta belajar yang lain. Pengujian ini diterapkan terhadap dua arsitektur jaringan, masing-masing dengan jumlah sel lapisan tengah 30 dan 40. Tabel 6-1 dan 6-2 memperlihatkan hasil pengujian masing-masing untuk jumlah sel 30 dan 40.

SR menunjukkan banyak pola yang dikenali dengan benar. FR banyak pola

yang dikenali dengan salah, yaitu pengenalan mengarah ke pola karakter lain, atau tidak dikenali sama sekali.

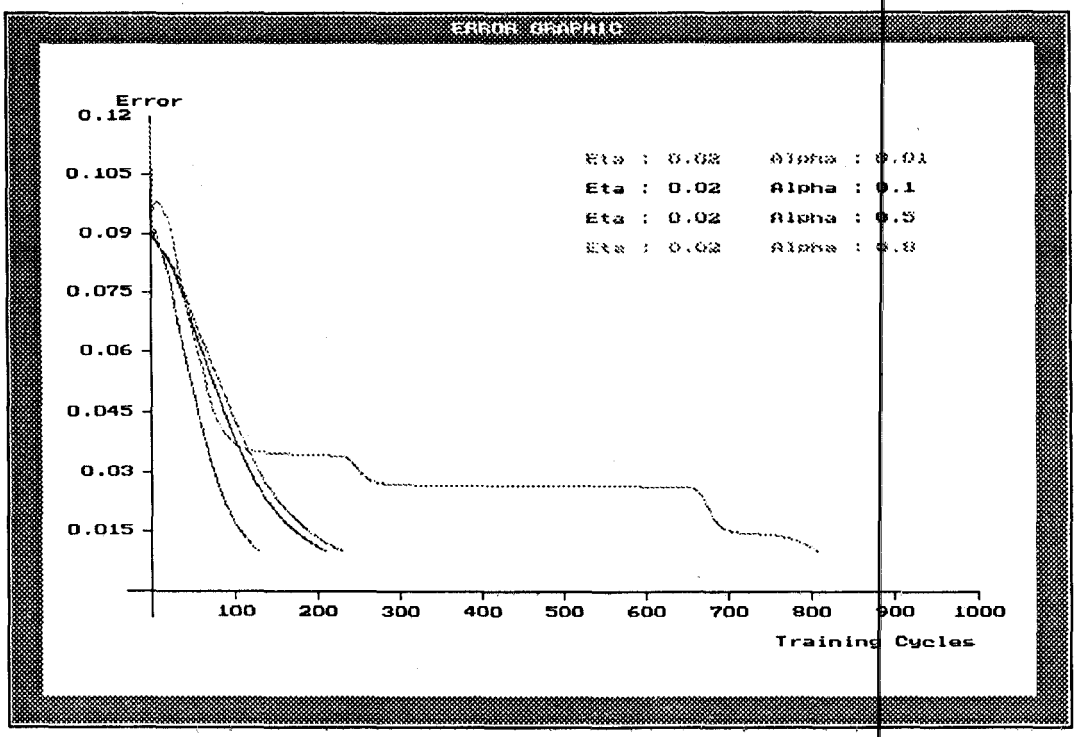
Tabel 6-1
Pengujian arsitektur 256-30-10

η	α	Error	E_{test}	Iterasi	SR	FR
0,01	0,0	0,009962	0,030829	464	27	2
	0,01	0,009980	0,030844	459	27	2
	0,1	0,009954	0,030781	418	27	2
	0,5	0,009957	0,030414	238	27	2
	0,6	0,009993	0,029871	198	27	2
	0,8	0,009969	0,026850	296	25	4
0,15	0,0	0,009980	0,030755	309	27	2
	0,01	0,009976	0,030746	306	27	2
	0,1	0,009940	0,030658	279	27	2
	0,5	0,009883	0,029789	164	27	2
	0,6	0,009994	0,028758	143	27	2
	0,8	0,009988	0,025547	419	25	4
0,02	0,0	0,009961	0,030640	232	27	2
	0,01	0,009934	0,030608	230	27	2
	0,1	0,009895	0,030508	210	27	2
	0,5	0,009965	0,028972	128	27	2
	0,6	0,009904	0,027936	123	27	2
	0,8	0,009951	0,024056	808	24	5
0,05	0,0	0,009887	0,028437	101	27	2
	0,01	0,009976	0,028481	100	27	2
	0,1	0,009802	0,027829	96	28	1
	0,5	0,009952	0,031367	93	25	4
	0,6	0,009877	0,032150	101	25	4

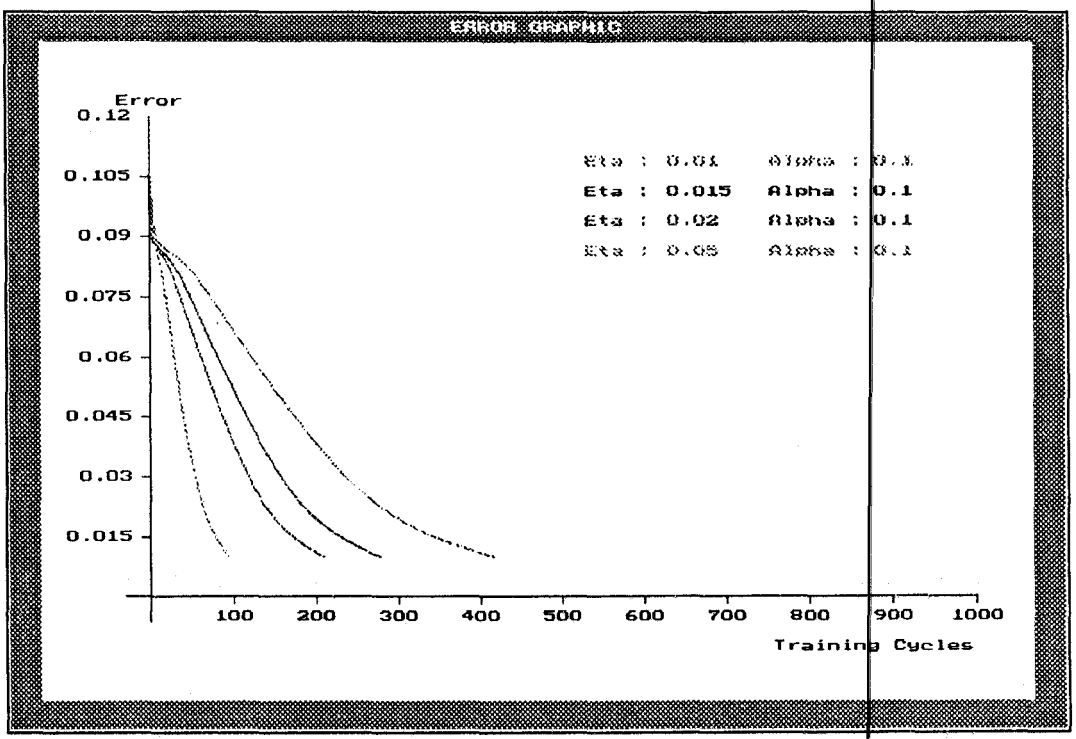
Tabel 6-2
Pengujian arsitektur 256-40-10

η	α	Error	E_{test}	Iterasi	SR	FR
0,02	0,1	0,009911	0,034596	186	25	4
	0,5	0,009861	0,034368	114	25	4
0,05	0,1	0,000149	0,034015	107	25	4
	0,5	0,000144	0,029849	496	25	4
0,15	0,005	0,000144	0,0151	600	28	1
	0,075	0,000148	0,0150	980	28	1
0,14	0,005	0,000149	0,0153	510	28	1
	0,075	0,000148	0,0151	630	28	1

Tabel 6-1 merangkum hasil belajar jaringan dengan jumlah sel dalam lapisan tengah sebanyak 30. Di sini terlihat pengaruh ditambahkannya metode momentum. Dengan memberikan konstanta momentum α yang semakin besar diperoleh konvergensi yang semakin cepat. Tetapi untuk tiap harga konstanta belajar η terdapat harga batas α, dimana di atas batas ini konvergensi menjadi lebih lama dan walaupun error tes lebih kecil tetapi jumlah pola yang dikenali berkurang. Hal ini karena jaringan kuat pada sebagian pola-pola namun untuk beberapa pola yang lain respon jaringan jauh berkurang. Di sini juga terlihat bahwa semakin tinggi harga η semakin sedikit iterasi yang diperlukan, dan harga batas α menurun. Tetapi η pun mempunyai harga batas, harga η yang terlalu tinggi dapat menyebabkan keakuratan jaringan berkurang, η yang terlalu rendah menyebabkan perhitungan menjadi lama.



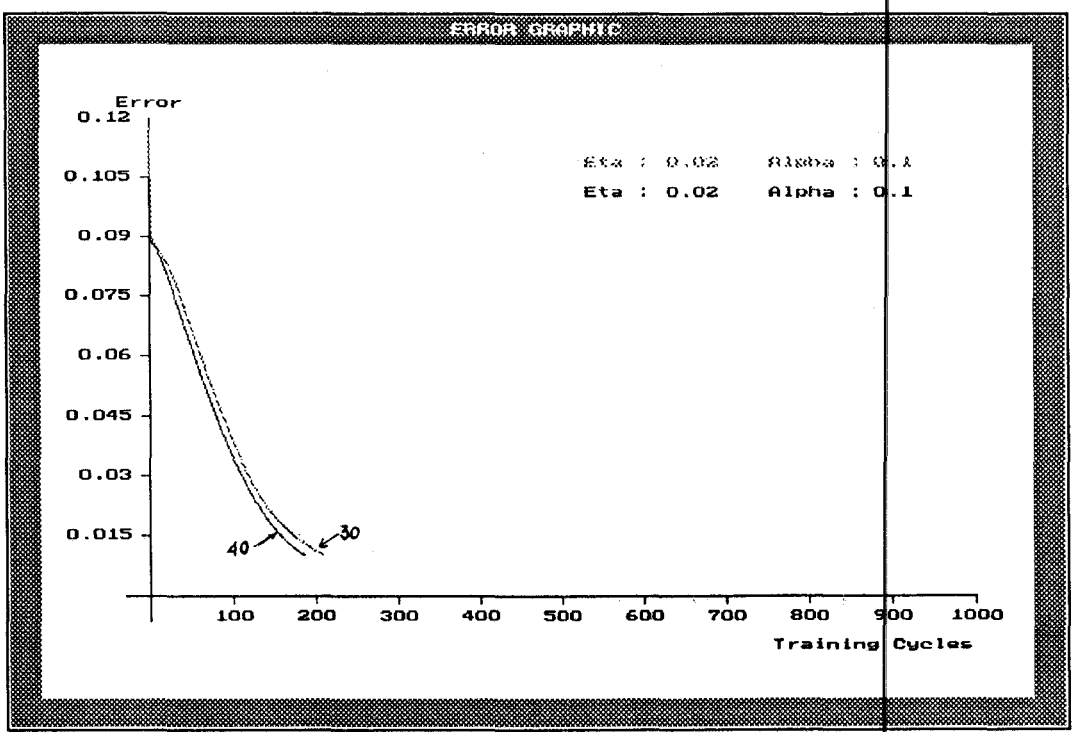
Gambar 6-3
Pengaruh momentum pada proses belajar



Gambar 6-4
Pengaruh konstanta belajar pada proses belajar

Gambar 6-3 memperlihatkan visualisasi pengaruh dipakainya metoda momentum dalam proses belajar. Semakin besar konstanta momentum penurunan error semakin cepat, namun α yang terlalu besar membuat konvergensi lambat.

Gambar 6-4 memperlihatkan visualisasi pengaruh kenaikan harga konstanta belajar η terhadap jumlah iterasi. η yang lebih besar membuat konvergensi berjalan cepat. Sedangkan Gambar 6-5 memperlihatkan visualisasi perbandingan arsitektur jaringan antara jumlah sel lapisan tengah 30 dengan 40 sel.



Gambar 6-5
Perbandingan arsitektur 256-30-10 dan 256-40-10

6.3 Pengujian Sistem

Setelah masing-masing bagian dari sistem yaitu perangkat keras pengambilan data dan jaringan saraf tiruan diuji, pengujian dilanjutkan dengan

mengaplikasikan langsung sistem untuk mengetahui prosentase kebenaran dari sistem. Arsitektur jaringan yang diuji di sini adalah 256-40-10 dengan konstanta belajar $\eta=0,15$ dan $\alpha=0,075$. Pola-pola yang dipakai, yang merupakan hasil tulisan tangan yang belum pernah diberikan pada jaringan sebelumnya, diperlihatkan pada Gambar 6-6. Hasilnya, dari 29 pola yang diberikan sebanyak 26 pola dikenali dengan benar (89,7%).

Gambar 6-6
Pola-pola untuk pengujian

BAB VII

P E N U T U P

7.1 Kesimpulan

Dari hasil proses perencanaan, pembuatan dan pengujian sistem dalam tugas akhir ini dapat ditarik beberapa kesimpulan :

1. Jaringan saraf tiruan mempunyai kemampuan yang dapat diandalkan sebagai satu alternatif pemecahan masalah pengenalan pola dengan jalan menemukan sendiri fungsi transfernya.
2. Untuk suatu arsitektur jaringan tertentu dibutuhkan pemilihan parameter-parameter belajar yang tepat.
3. Pemilihan parameter yang tepat akan mempercepat konvergensi tanpa mengurangi performa jaringan, dan sebaliknya pemilihan yang kurang tepat dapat mempercepat konvergensi namun tidak menjamin dihasilkannya performa yang bagus.
4. Selama proses belajar, sangat penting untuk selalu memonitor error; plot error yang turun dengan drastis dan kemudian cenderung stabil menandakan pemilihan parameter kurang tepat.
5. Banyak dan variasi pola latihan yang dipelajari menentukan kemampuan jaringan dalam mengklasifikasikan pola inputnya.
6. Pemrosesan awal sangat menetukan keberhasilan dari proses belajar jaringan saraf tiruan.

7. Transputer memberikan kemudahan dalam pemrograman parallel.
8. Dalam sistem transputer dengan basis PC, Transputer T805 dengan fasilitas unit floating-point dan link komunikasinya dapat digunakan dan berfungsi dengan baik.

7.2 Saran

Untuk pengembangan lebih lanjut penulis memberikan saran-saran sebagai berikut:

1. Penggunaan sensor jenis lain yang memperbaiki resolusi scanner, misalnya RAM optik, akan memungkinkan peralatan dapat diterapkan secara nyata.
2. Untuk proses yang lebih kompleks lagi dapat digunakan lebih dari satu tranputer.

DAFTAR PUSTAKA

1. Beale, R., ***Neural Computing : An Introduction***, IOP Publishing Ltd., Philadelphia, 1990.

2. Cok, Ronald S., ***Parallel Programs for the Transputer***, Prentice Hall Englewood Cliffs, New Jersey, 1991.

3. Dayhoff, Judith E., ***Neural Network Architectures : An Introduction***, Van Nostrand Reinhold, New York, 1990.

4. Eberhart, Russel J., dan Roy W. Dobbins, ***Neural Network PC Tools Practical Guide***, Academic Press Inc., San Diego, 1990.

5. Freeman, James A. dan David M. Skapura, ***Neural Network : Algorithms, Applications and Programming Techniques***, Addison-Wesley Publishing Company, 1991.

6. Graham, Ian dan Tim King, ***The Transputer Handbook***, Prentice Hall International (UK) Ltd., 1990.

7. Hall, Douglas V., ***Microprocessor and Interfacing : Programming and Hardware***, McGraw-Hill, Singapore, 1986.

8. INMOS Limited, ***Transputer Technical Notes***, Prentice Hall International (UK) Ltd., 1989.

9. Lippmann, Richard P., ***An Introduction to Computing with Neural Nets***, IEEE ASSP Magazine, April 1987.

10. Wasserman, Philip D., ***Neural Computing : Theory and Practice***, Van Nostrand Reinhold, New York, 1989.

11. Zurada, Jacek M., ***Introduction to Artificial Neural Systems***, Info Access Distribution Pte Ltd., Singapore, 1992.

12. ---, ***Transputer Education Kit : User Guide***, Computer System Architects, USA, 1990.

13. ---, ***Transputer Education Kit : Workbook***, Computer System Architects, USA, 1990.

14. ---, ***Transputer Architecture and Overview***, Computer System Architects, USA, 1990.

15. ---, ***Transputer Technical Specifications***, Computer System Architects, USA, 1990.

16. ---, ***Logical Systems C for The Transputer : Version 89.1 User Manual***, Computer System Architects, USA, 1990.